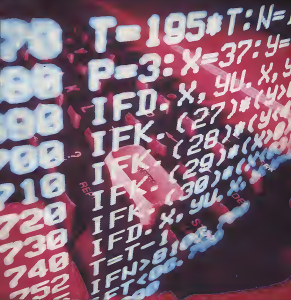
Prof. dr Nedeljko Parezanović

basic za galaksiju



Mikroračunarski sistem GALAKSIJA se sastoji od:

- mikroračunara,
- tastature,
- ekrana i
- kasetofona

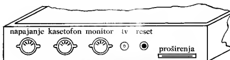
Mikroračunar se sastoji od mikroprocesora Z80A i operativne memorije. Operativna memorija se sastoji od ROM memorije u kojoj se nalaze sistemski programi i BASIC-interpretator. To je memorija kapaciteta 4 KB, sa mogućnošću proširenja za još 4 KB. Za programe korisnika na raspolaganju je RAM memorija kapaciteta do 6 KB u osnovnoj varijanti, a postoji mogućnost njenog proširenja do 54 KB. Tastatura je standardna tastatura sa 54 tastera. *Ekran* može biti standardni TV prijemnik ili crno beli-monitor. Na ekranu se tekst ispisuje u 16 redova, a svaki red može imati 32 znaka. Kada se vrši crtanje na ekranu, tada se na ekranu može prikazati najviše 64×48=3.072 tačke. Kao spoljna memorija koristi se standardni kasetofon sa brzinom upisa i čitanja od 280 bauda. Sistem ima mogućnost proširenja priključivanjem različitih uređaja preko posebnog priključka koji se nalazi na kutiji mikroračunara. Ukratko, tehničke karakteristike mikroračunarskog sistema GALAKSIJA su:

- mikroprocesor: Z80A, dužina reči 8 bita, osnovna učestanost 3,072 MHz,
- operativna memorija
  - — ROM od 4 ili 8 KB
  - — RAM do 6 KB, sa mogućnošću proširenja do 54 KB,
- ulazni uređaj:
  - — standardna tastatura sa 54 tastera,
- izlazni uređaj:
  - — TV prijemnik ili monitor, sa 16 redova od po 32 znaka u redu za prikaz teksta i 64×48 tačaka za grafički rad,
- spoljna memorija
  - — standardni kasetofon sa brzinom upisa i čitanja od 280 bauda,
- priključak za eventualna proširenja sistema,
- napajanje: 220V/4W

Programski sistem se sastoji od BASIC-interpretatora i sistemskih programa koji podržavaju rad sa periferním uređajima. Ovi programi su smešteni u ROM memoriju od 4 KB. U sistemu je predviđena mogućnost proširenja ROM memorije sa još 4 KB u koju bi se smestilo eventualno proširenje programskog sistema.

1.2. Povezivanje sistema i puštanje u rad

Mikroračunar GALAKSIJA smešten je u istu kutiju sa tastaturom. Na zadnjoj strani ove kutije nalaze se priključci za ostale uređaje u sistemu (sl. 1.2). Iz natpisa iznad svakog priključka jasna je namena priključaka. Pored priključaka nalazi se i jedan taster označen sa RESET. Pritisak



Sl. 1.2. Priključci na kutiji mikroračunara

na taster proizvodi inicijalizaciju sistema, ali ne briše program i podatke korisnika u RAM memoriji. Koristi se u izuzetnim situacijama kada sistem ne prihvata komande preko tastature, tada pritiskom na ovaj taster sistem se može »oporaviti« i dovesti u radno stanje, a da pri tome sadržaj memorije nije uništen. Ako ovo ne dovede sistem u radno stanje treba ga isključiti iz električnog napajanja i ponovo uključiti. Ovo će sistem dovesti u radno stanje, ali će sadržaj RAM memorije biti uništen.

Povezivanje sistema je prikazano na sl. 1.3. Dobro je povezivanje izvršiti sledećim redosledom:

- ako se koristi televizor tada treba povezati izlaz iz računara označen sa TV i antenski ulaz u televizor. Ako se koristi video monitor, tada treba izlaz iz računara, označen sa MONITOR, povezati sa video ulazom u monitor (veza 1),
- povezati izlaz iz ispravljača sa priključkom za napajanje na kutiji mikroračunara (veza 2),
- povezati ispravljač sa električnom mrežom napona 220V (veza 3),
- uključiti TV prijemnik, odnosno monitor, u električno napajanje (veza 4).

Kada je ovako povezan sistem, sveltosna indikacija na kutiji mikroračunara biće osvetljena. Ako se koristi monitor tada na ekranu monitora treba da se pojavi poruka sistema:

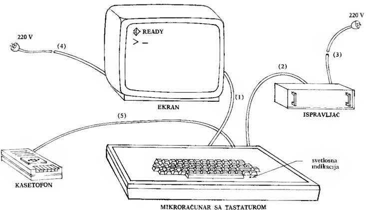
READY
> _

Engleska reč READY ukazuje da je sistem spreman za rad, a znak > ukazuje na red na ekranu u kojem će biti ispisana vaša poruka sistemu. Povlaka (_) predstavlja pokazivač pozicije na ekranu u koju će biti ispisan znak koji se unese sa tastature. U engleskoj literaturi ovaj pokazivač se zove *kursor*. Dakle, sistem je spreman za rad i može se unositi program ili se mogu zadavati druge raspoložive komande, koje ćemo upoznati u okviru ove knjige. Ako radite sa televizorom tada morate na UHF području pronaći 36. kanal i takođe ćete dobiti gore navedenu poruku sistema na ekranu.

Ako se koristi kasetofon tada treba povezati kasetofon sa priključkom na kutiji mikroračunara označenim sa KASETOFON (veza 5). Kasetofon raspolaže sa tri priključka:

MIC — mikrofonski ulaz,
EAR — zvučni ulaz i
REMOTE — upravljanje pokretanjem trake.

Sl. 1.3. Povezivanje mikroračunarskog sistema »Galaksija«



REMOTE priključak se ne koristi, pa ga ne treba ni povezivati. Međutim, vodite računa da ne priključite pogrešno mikrofonski ulaz i zvučni izlaz. Zato kabl za mikrofonski ulaz treba da ima posebnu oznaku. Rad sa kasetofonom, u cilju snimanja i čitanja programa, biće posebno objašnjen pri objašnjenju komandi SAVE i OLD. Ovde ćemo samo navesti da kasetofon ne mora biti nekog posebnog kvaliteta, a takođe i kasete. Međutim, dobro je da na kasetofonu postoji brojčanik. Na ovaj način lako ćete pronalaziti delove trake koji vas interesuju. Što se tiče kaseta bolje je koristiti kasete sa manjim trajanjem (na primer 60 min a ne 90 min.).

Ako raspolažete demonstracionom kasetom, tada možete lako učitati demonstracioni program i izvršiti ga. Za ovo treba uraditi sledeće:

- postaviti demonstracionu kasetu u kasetofon, ali voditi računa da je traka premotana tako da se čita od početka,
- preko tastature unesite komandu OLD i pritisnite taster za unošenje ENTER,
- pritisnite na kasetofonu taster za reprodukciju trake (najčešće označen sa PLAY),
- za vreme čitanja trake slika na ekranu će biti uklonuta. Kada je čitanje trake završeno slika se vraća i dobijate novu poruku READY i sistem očekuje vašu sledeću poruku,
- demonstracioni program je prenet sa trake u RAM memoriju i vašim zadavanjem komande RUN i pritiskom na taster ENTER počeće izvršavanje demonstracionog programa.

Cela ova procedura daje sledeću sliku na ekranu:

```
⊰> READY
 > OLD
⊰> READY
 > RUN
```

U slučaju da pri čitanju trake računar izda poruku WHAT? to je znak da traka nije korektno pročitana. U ovom slučaju treba smanjiti nivo izlaza iz kasetofona. Ovo se radi pomoću potenciometra za jačinu zvuka. Postoji i mogućnost da čitanje trake neograničeno dugo traje bez ikakvog izveštaja, tada treba pritisnuti RESET, pojačati nivo iz kasetofona i ponoviti čitanje trake. Može se dogoditi da ovo podešavanje treba više puta ponoviti sve dok se program korektno ne unese u memoriju. Kada je jedanput izlaz iz kasetofona podešen ne treba ga više dirati i ubuduće neće biti teškoća sa čitanjem trake.

Kada ste uneli poruku RUN i pritisli na taster ENTER, dalji rad računara je pod kontrolom demonstracionog programa. Kada se završi demonstracioni program sistem će ponovo doći u režim očekivanja vaše nove komande. Kad god ste vi na potezu računar će na ekranu izdati poruku READY ili će se nalaziti znak veće (>) na početku reda kojim vam računar ukazuje da treba da saopštite komandu.

Prema tome, vladaćete sistemom potpuno samo ako znate da izdajete prave komande u pravom trenutku. Cilj ove knjige je da vas obuči da komandujete računaru GALAKSIJA. Za ovo komandovanje mora se znati poseban programski jezik, u ovom slučaju BASIC-jezik.

2
INTERPRETATOR BASIC-JEZIKA

BASIC je programski jezik na kojem korisnik piše programe posredstvom kojih rešava svoje probleme na računaru. Kako je to jezik različit od mašinskog jezika mikroprocesora Z80A, to poseban program služi za tumačenje programa korisnika i proizvođenje odgovarajućih aktivnosti u sistemu. Ovaj program se zove *interpretator* BASIC-jezika i nalazi se u ROM memoriji pa je na taj način stalno na raspolaganju korisniku.

2.1. Struktura BASIC-programa

Program je tekst na programskom jeziku, u ovom slučaju BASIC-u, kojim korisnik računara opisuje postupak za rešavanje određenog problema na računaru. Analizom ovakvog teksta (programa) računar proizvodi niz aktivnosti koje daju rešenja postavljenog problema. Da bi korisnik sastavio program mora dobro da poznaje pravila pisanja pojedinih konstrukcija u programskom jeziku (sintaksu ili gramatiku jezika), kao i značenje svake konstrukcije (semantiku jezika). Osnovna konstrukcija u programskom jeziku, koja odgovara pojmu rečenice u prirodnom jeziku, jeste naredba. *Naredba* je konstrukcija u programskom jeziku koja ima određeno značenje i proizvodi određenu akciju u računarskom sistemu. Tako, na primer, naredba

PRINT "DANAS JE SUBOTA"

izdaje tekst DANAS JE SUBOTA na ekranu. Reč PRINT određuje akciju koju računar treba da izvrši, u ovom slučaju da izda tekst koji je naveden između znakova navoda. Naravno, računar ne analizira tačnost teksta koji izdaje, pa će tekst DANAS JE SUBOTA biti izdat i u nedelju i bilo koji drugi dan u nedelji! Program po pravilu sadrži veći broj naredbi koje se u tačno određenom redosledu moraju izvršavati na računaru. Ovaj redosled je obezbeđen tako što svaka naredba dobija broj koji se navodi ispred naredbe. Naredbe se ređaju u rastućem redosledu ovih brojeva. Tako, na primer, program

```
10 INPUT A
20 PRINT A
30 STOP
```

sadrži 3 naredbe označene brojevima 10, 20 i 30. U prvoj naredbi se saopštava računaru da prihvati broj sa ulaza (tastature) i dodeli promenljivoj A, u drugoj da izda ovaj broj na ekranu i u trećoj da završi rad (naredba STOP)

Gornji tekst programa se sastoji od 3 reda koje ćemo zvati *programski redovi*, a broj koji ukazuje na redosled redova zvaćemo *broj reda*. Kao što ćemo videti kasnije, programski red može biti i bez broja reda. Da bismo razlikovali programske redove koji sadrže broj reda, od onih koji ovaj broj ne sadrže, prve ćemo zvati *označeni programski redovi*, a druge *neoznačeni programski redovi*. Prema tome, BASIC-*program* je jedan ili više označenih programskih redova. BASIC-program se čuva u RAM memoriji. Memorijski prostor u kojem se čuva program zvaćemo *zona programa*. Program, u zoni programa, se izvršava pomoću komande RUN. Tako sledeća slika na ekranu prikazuje postupak unošenja i izvršavanja gore navedenog programa:

```
> 10 INPUT A
> 20 PRINT A
> 30 STOP
> RUN
 ? 132
 132
> _
```

Kao što se vidi, kada je saopštena komanda RUN počelo je izvršavanje programa i računar je izdao znak pitanja (?) što ukazuje korisniku da treba da unese brojni podatak koji će se u naredbi INPUT dodeliti promenljivoj A. U ovom slučaju otkucan je broj 132 i pritisnut taster ENTER posle čega je ovaj broj unet u računar i dodeljen promenljivoj A. Sledeća naredba PRINT izdaje ovaj broj na ekranu i program se završava. Pošto se program nalazi u memoriji to ga možete proizvoljan broj puta izvršiti, koristeći komandu RUN.

Programski red može imati veći broj naredbi. U ovom slučaju naredbe se među sobom razdvajaju sa dve tačke (:). Tako se gornji program može uneti u memoriju i izvršiti u sledećem obliku:

```
> 10 INPUT A: PRINT A: STOP
> RUN
 ? 132
 132
> _
```

Na kraju da navedemo još jednu mogućnost. Neoznačeni programski redovi se izvršavaju neposredno po unošenju. Međutim, ovakvi redovi se ne čuvaju u memoriji računara pa se ne može ponoviti njihovo izvršavanje primenom komande RUN. Korišćenjem neposrednog izvršavanja naredbi možemo upotrebiti računar za manja izračunavanja za koja bi inače koristili kalkulator. Tako, ako želimo da pomnožimo brojeve 423 i 127, to možemo uraditi na sledeći način:

```
> PRINT 423*127
 53721
```

Dakle, unošenje naredbe PRINT u neoznačenom programskom redu i navođenje aritmetičkog izraza dovodi do izračunavanja vrednosti aritmetičkog izraza i izdavanja rezultata na ekranu. U ovom slučaju to je broj 53721. Naredba INPUT se ne može koristiti u režimu neposrednog izvršavanja. Ovo nije značajno ograničenje.

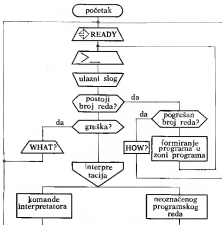
2.2. Struktura BASIC-interpretatora

Već smo videli da se posle uključivanja računara na ekranu dobija tekst READY i znak veće (>) na početku reda. Računar je pod kontrolom BASIC-interpretatora i očekuje našu poruku. Tekst koji vi saopštavate računaru zvaćemo *ulazni slog*. Ovaj tekst može imati najviše 125 znakova i na kraju ovog teksta morate pritisnuti taster ENTER, nakon čega će tekst biti prihvaćen od strane BASIC-interpretatora i počeće njegova analiza. Dakle, ulazni slog ne odgovara dužini reda na ekranu! Analizom teksta BASIC-interpretator će utvrditi da uneti tekst predstavlja:

- označen programski red,
- neoznačen programski red,
- komandu BASIC-interpretatora ili
- nerazumljivi tekst.

Ako ulazni slog predstavlja označen programski red, tada će ovaj biti unet u zonu programa. Ako je ulazni slog neoznačen programski red onda će ovakav programski red biti odmah izvršen i neće biti unet u zonu programa kao deo programa. Neka saopštenja korisnika nisu naredbe BASIC-jezika, već se odnose na rad BASIC-interpretatora, pa ćemo ih zvati *komandama*. Tako, RUN je komanda BASIC-interpretatora kojom otpočinje izvršavanje programa u zoni programa. Prema tome, sadržaj ulaznog sloga može biti komanda i ova se neposredno izvršava. Komande ne mogu biti sastavni deo programskih redova. Sve komande BASIC-interpretatora upoznaćemo u ovom delu knjige. Ako analiza ulaznog sloga pokaže da tekst ne predstavlja programski red (označen ili neoznačen) niti komandu, onda je takav tekst nerazumljiv BASIC-interpretatoru i ovaj će izdati poruku WHAT? što ukazuje da je to tekst koji se ne može prihvatiti (greška).

Na sl. 2.2.1 prikazana je blok-shema BASIC-interpretatora. Ako pratimo ovu shemu vidimo da će najpre (na početku rada) biti izdata poruka READY i znak veće (>). Sistem sada očekuje ulazni slog. Korisnik unosi tekst i pritiska taster ENTER. Sada BASIC-interpretator ispituje da li uneti tekst počinje brojem. Ako počinje brojem onda se ispituje da li je to broj iz intervala [1,32767]. Ako broj pripada navedenom intervalu, onda je to *broj reda* i ulazni slog se uzima kao označen programski red i smešta se u zonu programa radi formiranja programa. Ako uneti broj nije iz navedenog intervala, onda je to greška i sistem izdaje poruku HOW? i READY a zatim očekuje novi ulazni slog. Ako je ulazni slog komanda ili neoznačen programski red, tada se vrši interpretacija (izvršavanje) odgovarajuće komande ili naredbe BASIC-jezika. Kada je izvršavanje



Sl. 2.2.1. Blok-shema BASIC — interpretatora

završeno ponovo se očekuje poruka korisnika. Ako ulazni slog sadrži tekst koji nije mogao biti protumačen kao programski red (označen ili neoznačen) niti kao komanda, tada sistem javlja grešku sa izveštajem WHAT? i očekuje novu poruku korisnika. Ostaje nam da upoznamo komande interpretatora i naredbe BASIC-jezika, pa ćemo moći uspešno da koristimo računar GALAKSIJU.

Primer.

Ako računaru saopštimo poruku u obliku označenog programskog reda, onda će ovakav red biti poslat u zonu programa i komandom RUN može se izvršiti. Na primer:

```
> 10 PRINT "DANAS JE SUBOTA"
> RUN
DANAS JE SUBOTA
> _
```

Možemo istu naredbu uneti u obliku neoznačenog programskog reda i biće neposredno izvršena, tj.

```
> PRINT "DANAS JE SUBOTA"
DANAS JE SUBOTA
> _
```

Međutim, ako računaru saopštimo ulazni slog u obliku:

```
> DANAS JE SUBOTA
WHAT?
```

tada ovo neće biti prihvaćeno, jer ovakvu poruku računar ne razume, pošto nije programski red niti komanda interpretatora. Takođe, poruka u obliku:

```
> 32768 PRINT "DANAS JE SUBOTA"
HOW?
```

neće biti prihvaćena, jer sadrži nedozvoljen broj reda.

2.3. Komande interpretatora

Korišćenje računara posredstvom BASIC-jezika ima i n t e r a k t i v n i karakter, što znači da u toku izvršavanja programa korisnik komunicira sa programom izmenjujući različite poruke. Međutim, korisniku je neophodno da na lak način izvodi i izvesne manipulacije sa programom, kao što je unošenje programa u zonu programa, ispravke programa, izvršavanje programa i slično. Ovo se postiže posredstvom poruka koje daje korisnik sistemu, a koje smo nazvali k o m a n d a m a interpretatora. Komande ne mogu biti sastavni deo programa, pa i ulazni slog preko kojeg se saopštavaju računaru ne može da sadrži broj reda. Sve komande se neposredno izvršavaju. Komanda se piše kao odgovarajuća engleska reč i pored ovakvog zapisa, mogu se skraćivati, tako, što se piše samo početno slovo reči iza kojeg sledi tačka. U ovom odeljku upoznaćemo komande BASIC-interpretatora za računar GALAKSIJU.

2.3.1. Unošenje novog programa

Pre nego što se počne unošenje novog programa korisnik mora biti siguran da već ne postoji program u zoni programa. Zato pre unošenja novog programa treba primeniti komandu

NEW

Ova komanda briše tekući program u zoni programa.

Primer.

Unećemo ranije izložen program, ali tako što ćemo najpre komandom NEW izbrisati zonu programa, tj.

```
> NEW
> 10 INPUT A
> 20 PRINT A
> 30 STOP
> _
```

Program od 3 programska reda nalazi se u zoni programa i sada možemo primeniti komandu za izvršavanje programa.

2.3.2. Ispravke programa

Unošenje programa preko tastature vrlo je podložno greškama. Zato moraju postojati načini kako korisnik može ispraviti nastalu grešku. Za ovo postoje sledeće mogućnosti:

Brisanje poslednjeg unetog znaka

Ako korisnik primeti, pri kucanju, da je pogrešan poslednji otkucan znak, ovo može ispraviti pritiskom na taster označen strelicom nalevo (←). Pritiskom na ovaj taster briše se poslednji otkucan znak i kursor pomera za jedno mesto ulevo. Na ovaj način korisnik može izbrisati i veći broj znakova u ulaznom slogu, sleva nadesno, ako više puta pritisne taster označen strelicom nalevo.

Brisanje tekućeg ulaznog sloga

Ako korisnik želi da izbriše ceo ulazni slog, pre nego što je pritisnut taster ENTER za unošenje sloga, to može učiniti pritiskom na taster za brisanje jednog znaka (←) i na taster za ponavljanje (označen sa REPT). Taster REPT ponavlja funkciju poslednjeg pritisnutog tastera onoliko dugo koliko se drži pritisnut. Ovo što smo naveli, za brisanje tekućeg ulaznog sloga, je samo jedna korisna funkcija ovog tastera. Brisanje tekućeg ulaznog sloga može se izvršiti tastaturnom komandom BRK (vidi odeljak 2.4).

Izbacivanje programskog reda iz zone programa

U zoni programa programski redovi su označeni i uređeni po rastućim brojevima redova. Ako se želi izbaciti bilo koji programski red dovoljno je uneti ulazni slog koji sadrži samo broj reda koji se želi izbaciti. Tako, na primer, ako ulazni slog sadrži samo broj 30 i pritisnemo taster ENTER za njegovo unošenje, tada će iz zone programa biti izbačen programski red sa brojem reda 30.

Ubacivanje programskog reda u zonu programa

Ako želite ubaciti novi programski red, tada treba otkucati novi označen programski red i posle pritiska na

ENTER ovaj će biti unet u zonu programa i to na odgovarajuće mesto, tako da će programski redovi u zoni programa biti u propisanom rastućem redosledu brojeva redova. Tako, ako unesemo programski red sa brojem reda 15, a u zoni programa postoje programski redovi sa brojevima 10 i 20, tada će novi programski red biti umetnut između postojećih redova 10 i 20.

Zamena programskog reda u zoni programa

Ako se unese programski red sa brojem reda koji već postoji u zoni programa, tada će postojeći programski red biti zamenjen novim bez obzira na dužine ovih redova.

Izmene unutar programskog reda

Vrlo često korisnik ima potrebu da izmeni deo programskog reda koji se nalazi u zoni programa. U ovom slučaju treba koristiti posebnu komandu

EDIT n

gde je n broj programskog reda koji se želi menjati. Posle unošenja ove komande briše se ekran i u prvom redu na ekranu izdaje programski red iz zone programa sa brojem n. Kursor se nalazi na kraju programskog reda. Pritiskom na taster sa strelicom nalevo kreće se kursor nalevo, a pritiskom na taster sa strelicom nadesno kursor se kreće nadesno. Na ovaj način pozicionirate kursor u okviru programskog reda, ali ne vršite brisanje znakova. Ubacivanje novog teksta postižete jednostavnim kucanjem teksta koji ste želeli ubaciti na odabranu poziciju kursora. Pritiskom na taster DEL izbacujete znake desno od kursora. Kada ste izvršili željene ispravke pritisnete taster ENTER i ispravljen red ide na odgovarajuće mesto u zoni programa. Naravno, na ovaj način možete promeniti i broj reda, pa postaviti novu naredbu u program. Samo vodite računa da dovođenjem naredbe iz zone programa, sa EDIT, radi ispravke, nije ova naredba izbačena iz zone programa. Ona će biti zamenjena ispravljenom naredbom, ako je ostao isti broj reda, tek posle pritiska na taster ENTER. Ako ste se u toku ispravljanja naredbe predomislili, pa želite da ostavite postojeću naredbu u programu, a proces ispravljanja da prekinete treba da pritisnete taster BRK, što će vratiti sistem na očekivanje nove poruke korisnika.

2.3.3. Izdavanje programa

Često korisnik želi da vidi izgled programa u zoni programa. Ovo može postići na dva načina: korišćenjem komande LIST ili tastera sa oznakom LIST.

Korišćenje komande LIST

Ova komanda omogućuje izdavanje programa iz zone programa na ekran od navedenog broja reda. Komanda se piše u obliku:

LIST n

gde je n broj reda od kojeg počinje izdavanje na ekranu. Izdavanje se vrši samo za vreme dok se taster ENTER drži pritisnut. Kada se taster otpusti prestaje izdavanje, ali se novim pritiskom na taster nastavlja izdavanje. Ako se želi prekinuti dalje izdavanje treba pritisnuti taster BRK. Ako se broj reda (n) ne navede tada izdavanje počinje od početka programa, tj. od programskog reda sa najmanjim brojem reda.

Korišćenje tastera LIST

Izdavanje programa može se izvršiti i pritiskom na taster LIST. U ovom slučaju izdavanje uvek počinje od početka programa. Izdavanje traje samo dok se taster LIST drži pritisnut, kada se taster otpusti prestaje izdavanje. Ponovnim pritiskom izdavanje se nastavlja. Ako se želi prekinuti dalje izdavanje treba pritisnuti taster BRK.

2.3.4. Izvršavanje programa

Mi smo već upoznali komandu za izvršavanje programa (RUN). Međutim, ova komanda ima opšti oblik

RUN n

gde je n broj reda od kojeg počinje izvršavanje programa u zoni programa. Ako se broj reda ne navede, već samo reč RUN, tada će početi izvršavanje programa od početka.

2.3.5. Snimanje programa na kaseti

Korisnik često želi da program sačuva iz zone programa u spoljnoj memoriji — u našem slučaju na kaseti. Ovako sačuvan program može se po potrebi preneti sa trake (kasete) u zonu programa. Snimanje programa na kaseti vrši se komandom

SAVE

Ova komanda vrši prenošenje programa iz zone programa na kasetu, pri čemu program ostaje u zoni programa. Pri snimanju programa na kaseti postupite na sledeći način:

- postavite kasetu u kasetofon i premotajte traku na slobodan deo, ako već imate snimljenih programa na traci,
- pritisnite prekidače za snimanje na kasetofonu (obično označeni sa PLAY i RECORD),
- unesite komandu SAVE i pritisnite taster ENTER,
- za vreme snimanja slika na ekranu se ne izdaje, a kada se snimanje završi pojavice se slika i poruka sistema READY sa znakom (>) za očekivanje poruke korisnika,
- zaustavite kasetofon i zapišite na kaseti ime programa i stanje brojača koje pokazuje na kojem delu trake se nalazi snimljeni program.

Preporučljivo je napraviti dva snimka programa na traci, tako da u slučaju oštećenja trake jedan snimak bude sačuvan. Ovo možete uraditi tako što bez zaustavljanja trake posle prvog snimanja unesete još jedanputa komandu SAVE, ili ponavljanjem gornje procedure.

2.3.6. Unošenje programa sa kasete

Program koji se nalazi na kaseti prenosi se sa kasete u zonu programa pomoću komande

OLD

Ova komanda uništava prethodni program u zoni programa i upisuje novi program sa kasete. Pri čitanju programa sa kasete postupite na sledeći način:

- postavite u kasetofon kasetu na kojoj se nalazi program i pomoću brojača na kasetofonu premotajte traku na početak programa,
- unesite komandu OLD i pritisnite taster ENTER, pritisnite na kasetofonu taster za reprodukciju trake (najčešće označen sa PLAY),
- za vreme čitanja trake slika na ekranu se ne izdaje. Kada je čitanje programa završeno slika se vraća i poruka READY sa znakom (>) za očekivanje poruke korisnika,
- zaustavite kasetofon, program sa trake je prenet u zonu programa i može se izvršiti komandom RUN.

Ako iz bilo kojih razloga ne izvršite uspešno prenošenje programa sa kasete u zonu programa, tada ponovite gornji postupak. Takođe, imajte u vidu da je možda potrebno podesiti nivo signala iz kasetofona, kao što je opisano u odeljku 1.2.

2.3.7. Verifikacija snimljenog programa

Kada se izvrši snimanje programa na kaseti, korisnik može zaželeti da se uveri da li je snimanje izvršeno korekt-

no. Ovo može učiniti unošenjem snimljenog programa sa kasete u zonu programa. Međutim, ovaj postupak ima jedan krupan nedostatak, jer se pri unošenju programa u zoni programa uništava stari program. Zato je za ovo pogodnije koristiti komandu

OLD?

koja vrši poređenje programa sa trake (kasete) sa programom u zoni programa. Ako su oba programa ista dobija se poruka READY, a ako postoji razlika poruka WHAT? U slučaju poruke WHAT? treba izvršiti ponovno snimanje programa iz zone programa, a ovaj je posle verifikacije sačuvan u zoni programa. Za verifikaciju treba primeniti sledeću proceduru:

- posle izvršenog snimanja programa vratiti traku na početak snimljenog programa,
- uneti komandu OLD? i pritisnuti taster ENTER,
- pritisnuti na kasetofonu prekidač za reprodukciju trake (PLAY),
- za vreme verifikacije slika na ekranu se ne izdaje. Ako je po završenoj verifikaciji dobijena poruka READY program je ispravno snimljen, a ako se dobije poruka WHAT? otkrivena je greška, pa snimanje treba ponoviti.

2.4. Tastaturne komande

Neke komande se mogu aktivirati direktno pritiskom na odgovarajuće tastere na tastaturi (sl. 2.4.1). Ovakve komande ćemo zvati *tastaturne komande*.

Brisanje ekrana i unetog znaka

SHIFT/DEL — briše ekran i postavlja kursor na početak prvog reda na ekranu,

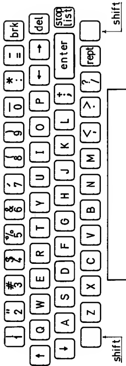
← — briše poslednji uneti znak u tekućem ulaznom slogu

Unošenje naredbi i komandi

ENTER — unosi tekući ulazni slog i postavlja kursor na početak sledećeg reda

Izdavanje programa

LIST — izdaje program iz zone programa na ekran za vreme dok se taster drži pritisnut



Sl. 2.4.1. Tastatura računara »GALAKSIJA«

Ispravke naredbi komandom EDIT

← — pomera kursor za jedno mesto ulevo,

→ — pomera kursor za jedno mesto udesno,

DEL — briše znak desno od kursora,

ENTER — završava ispravljanje naredbe i ispravljenu naredbu unosi u zonu programa.

Višestruko unošenje

REPT — ponavlja poslednji uneti znak za vreme dok je taster pritisnut.

Komande od značaja za vreme izvršavanja programa

DEL — suspenduje izvršavanje programa za vreme dok je taster pritisnut,

BRK — prekida izvršavanje programa i izdaje izveštaj. BREAK n, gde je n broj reda naredbe koja treba da se izvrši kao sledeća naredba u programu. Unošenje programskog reda može se prekinuti pritiskom na ovaj taster u kom slučaju se izdaje samo reč BREAK (što znači prekid).

2.5. Izveštaji o greškama

Važan deo implementacije BASIC-jezika je izdavanje izveštaja o otkrivenim greškama u programskim redovima i komandama interpretatora koje korisnik zadaje. Ovo predstavlja veliku pomoć korisniku pri radu sa računarom. Greške mogu biti otkrivene pri radu sa BASIC-interpretatorom ili pri izvršavanju programa.

Greške pri radu sa interpretatorom

Ovo su greške koje mogu biti otkrivene pri analizi ulaznog sloga (vidi sl. 2.2.1.). To mogu biti sledeći izveštaji o greškama.

WHAT? — sintaksna greška koja ukazuje da ulazni slog nije komanda interpretatora niti naredba BASIC-jezika,

HOW? — ulazni slog počinje brojem koji se ne može prihvatiti kao broj reda.

SORRY — nedovoljno memorijskog prostora za registrovanje elemenata niza ili naredbi programa

Greške pri izvršavanju programa

Otkrivanje grešaka u programu je posebno značajno za korisnika, jer ovo olakšava ispravke programa i pomaže korisniku da dođe do korektnog programa. Pri izvršavanju programa mogu se dobiti sledeći izveštaji o greškama

WHAT? — otkrivena sintaksna greška u naredbi BASIC-programa. Pored ove reči izdaje se i označen programski red u kojem je otkrivena greška sa postavljenim znakom pitanja (?) na mestu otkrivanja greške. Na primer:

```
WHAT?
20 P?INT A
```

U ovom primeru reč PRINT je zapisana kao PINT, što je otkriveno kao sintaksna greška

HOW? — otkrivena greška u veličini brojnog podatka, prelaza na nepostojeći broj reda u programu i dr. Pored ove reči izdaje se i označen programski red u kojem je otkrivena greška sa postavljenim znakom pitanja (?) na mestu otkrivanja greške
Na primer:

```
HOW?
20 PRINT 1E40?*A
```

U ovom primeru je otkriven brojni podatak 1E40 koji ne može biti registrovan u memoriji računara.

SORRY — otkriven zahtev za memorijskim prostorom u RAM memoriji za koji ne postoji mogućnost s obzirom na raspoloživi kapacitet memorije Na primer

```
SORRY
20 PRINT A(I)?
```

U ovom primeru za niz brojnih podataka nije bilo dovoljno mesta u memoriji.

Izdavanje označenog programskog reda u kojem je otkrivena greška, omogućuje korisniku da lako pozove odgovarajući programski red komandom EDIT i izvrši potrebne ispravke.

Kratak izvod

U okviru ove glave upoznali smo BASIC-interpretator u onoj meri koliko je to neophodno da uspešno koristimo računar GALAKSIJU. Navodimo najvažnije informacije:

- tekst koji korisnik saopštava interpretatoru zovemo ulazni slog,
- ulazni slog može imati najviše 125 znakova,
- ulazni slog može biti:
  - — označen programski red,
  - — neoznačen programski red,
  - — komanda interpretatora,
  - — nerazumljivi tekst,
- BASIC-program čine jedan ili više označenih programskih redova,
- komande i neoznačeni programski redovi se izvršavaju neposredno,
- broj reda mora biti iz intervala [1,32767],
- komande interpretatora su:

  NEW — briše zonu programa,
  EDIT — uređenje označenog programskog reda,
  LIST — izdavanje programa,
  RUN — izvršavanje programa,
  SAVE — snimanje programa na kaseti,
  OLD — čitanje programa sa kasete,
  OLD? — verifikacija upisanog programa,

- sve komande interpretatora se mogu pisati u skraćenom obliku koji se sastoji samo od prvog slova komande za kojim sledi tačka. Tako na primer: umesto RUN može se pisati R., umesto LIST može se pisati L. itd.
- tastaturne komande se aktiviraju neposredno po pritisku odgovarajućeg tastera,
- izveštaji o greškama:

  WHAT? — sintaksna greška,
  HOW? — greška u veličini brojnog podatka i dr.,
  SORRY — nedovoljan kapacitet memorije

3
AZBUKA

Svaki pisani jezik se gradi nad odabranim skupom simbola. Tako su prirodni jezici izgrađeni nad skupom simbola koji se sastoji od slova, cifara i interpunkcijskih znakova. Ovakav skup simbola, nad kojim se gradi jezik, zovemo a z b u k o m. Za izgradnju veštačkih jezika, kakvi su i programski jezici, takođe je neophodno odabrati skup simbola koji će činiti azbuku jezika. Kako želimo da se programski jezik što manje razlikuje od prirodnog jezika, to za azbuku programskog jezika biramo iste simbole kao i u azbukama prirodnih jezika.

3.1. Osnovni simboli

Azbuka BASIC-jezika sastoji se od sledećih simbola:

1. Velika slova latinske azbuke: A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
2. YU-slova: Ć | Č | Ž | Š
3. Dekadne cifare: 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
4. Interpunkcijski znaci: ! | " | (|) | – | : | ; | , | . | ? | ␣
5. Simboli aritmetičkih operacija: + | — | * | /
6. Relacijski simboli: < | > | =
7. Pomoćni znaci: # | $ | % | &
8. Grafički simbol: ▦

gde je ␣ oznaka za prazninu (blanko).

Svi ovi simboli predstavljaju jedinstvene znake koji se ne mogu deliti na elementarnije delove. Zato ove simbole zovemo *osnovnim simbolima*. Svaki od navedenih osnovnih simbola unosi se u računar pritiskom na odgovarajući taster ili jednovremenim pritiskom na taster SHIFT i odgovarajući taster (sl. 2.4.1). Jednovremeni pritisak na taster SHIFT i taster x označavaćemo sa SHIFT/x. Taster SHIFT treba koristiti za unošenje sledećih znakova.

SHIFT/C unosi slovo Ć
SHIFT/X unosi slovo Č
SHIFT/Z unosi slovo Ž
SHIFT/S unosi slovo Š
SHIFT/7 unosi grafički simbol

Pored navedenih znakova SHIFT se mora koristiti pri unošenju svakog gornjeg znaka na tasteru na kojem postoje dva znaka. Na primer, SHIFT/2 unosi znak navoda (") itd. Svi ostali tasteri unose isti znak bez obzira da li je taster SHIFT pritisnut ili ne. Tako, SHIFT/A unosi slovo A, kao i pritisak na taster A bez tastera SHIFT.

3.2. Izvedeni simboli

Ako je potrebno uvoditi nove simbole, onda se ovi grade kao niske osnovnih simbola i zovemo ih izvedeni simboli. Izvedeni simboli su:

1. Izveden relacijski znak: EQ
2. Službene reči: ARR$ | BYTE | CALL | DOT | ELSE | FOR | GOTO | HOME | IF | INPUT | NEXT | PRINT | RETURN | STEP | STOP | TAKE | TO | UNDOT | WORD

Svaka od navedenih službenih reči koja sadrži više od dva slova može se skratiti, tako da se navede samo prvo slovo reči iza kojeg se piše tačka. Tako se umesto INPUT može pisati I. . Ovo će omogućiti brže unošenje programa preko tastature, ali i znatno manje angažovanje memorijskog prostora za čuvanje programa. Nedostatak ovakvog skraćivanja službenih reči jeste smanjena preglednost i čitljivost programa. Zato u primerima koji ilustruju naredbe BASIC-jezika nećemo primenjivati navedeno skraćivanje, već to ostavljamo čitaocu da primeni kada ovlada pisanjem BASIC-programa.

Službena reč		
Piše se	Čita se	Znači
ARR$	ərej	niz
BYTE	bajt	bajt
CALL	ko:l	poziv
DOT	dot	tačka
ELSE	els	inače
FOR	fo:	za
GOTO	gou tu:	idi na
HOME	houm	ka polazištu (kući)
IF	if	ako
INPUT	input	ulaz
NEXT	nekst	sledeći
PRINT	print	štampaj
RETURN	ritə:n	povratak
STEP	step	korak
STOP	stop	zaustavi
TAKE	teik	uzeti
TO	tu	do
UNDOT	ʌndot	brisati tačku
WORD	wə:d	reč

Tabela 3.I. Službene reči BASIC-jezika

Kao što vidimo, azbuka BASIC-jezika sadrži uobičajene tipografske znake, ali je proširena izvedenim simbolima. Osnovne simbole nije teško zapamtiti, jer ih već poznajemo, a izvedene simbole lako ćemo zapamtiti kroz njihovu primenu pri pisanju programa. Službene reči imaju dvostruku ulogu. Prvo, one ukazuju interpretatoru na funkciju određene konstrukcije BASIC-jezika. Drugo, kao reči engleskog jezika, svojim značenjem, omogućuju lako pamćenje funkcije naredbe od strane programera. Tako, službena reč STOP služi kao simbol koji u interpretatoru proizvodi kraj rada po programu. Međutim, ista reč svojim značenjem omogućuje čoveku — programeru lako pamćenje funkcije naredbe. U ovom slučaju reč znači »zaustaviti«, što jasno govori i o funkciji naredbe. Kao što se vidi korisno je znati značenje službenih reči u engleskom jeziku. Zato su u tabeli 3.1. date službene reči: njihov način pisanja, izgovor i značenje. Naravno, sa gledišta rada računara od značaja je samo tačno pisanje ovih reči. Greške u pisanju službenih reči proizvode prekid rada po programu sa izveštajem WHAT? i programskim redom u kojem je greška otkrivena.

Kratak izvod

- Azbuka BASIC-jezika sastoji se od osnovnih i izvedenih simbola.
- Osnovni simboli su uobičajeni tipografski znaci: slova, cifre i specijalni znaci.
- Izvedeni simboli su niske osnovnih simbola koje se moraju pisati na propisan način.
- Izvedeni simboli su izvedeni relacijski znak i službene reči.

4
OSNOVNE SINTAKSNE JEDINICE

Naredba BASIC-jezika je niska simbola nad azbukom BASIC-jezika. Međutim, kao što sve niske slova u našem pravopisu ne čine rečenice, tako i sve niske simbola BASIC-jezika ne čine naredbe. Da bismo na lak i pregledan način definisali niske simbola koje čine naredbe, pa i BASIC-program, uvodimo *sintaksne jedinice*. Ovo činimo na sličan način kao i u gramatikama prirodnih jezika gde uvodimo gramatičke kategorije, kao što su subjekat, predikat, imenica, glagol i sl., da bismo izrazili pravila kako se konstruišu rečenice u jeziku. Sintaksne jedinice delimo na elementarne i složene.

4.1. Elementarne sintaksne jedinice

Elementarne sintaksne jedinice služe za definisanje niski BASIC-jezika koje imaju određeno značenje, ali ne mogu same za sebe stajati u programu. Ovakve niske ćemo zvati *elementarne konstrukcije* u jeziku, što odgovara rečima u prirodnom jeziku, jer reči imaju određeno značenje, ali tek u okviru rečenice izražavaju određenu misao. Elementarne sintaksne jedinice su:

— podatak
— promenljiva,
— niz i
— izraz.

Zapis informacije koja je predmet obrade na računaru zove se *podatak*. Informacija koja se obrađuje na računaru može biti kvantitativne ili kvalitativne prirode. Kvantitativna informacija se piše kao *brojni podatak*, a kvalitativna kao *azbučni podatak*. U matematici je uobičajeno da se za podatak kaže *konstanta*. Mi ćemo koristi i jedan i drugi naziv. Kako podatak može biti brojni i azbučni to se i promenljive kojima se mogu dodeliti ovakvi podaci zovu *brojne i azbučne promenljive*. Više brojnih i azbučnih podataka koji imaju zajedničko ime zovu se *niz*. Ako se niz sastoji od brojnih podataka, tada se zove *brojni niz*, a ako se sastoji od azbučnih podataka zove se *azbučni niz*.

Izraz je zapis postupka za dobijanje jednog rezultujućeg podatka na osnovu jednog ili više zadatih podataka. Izraz može biti brojni, azbučni i relacijski. Detaljnije definicije elementarnih sintaksnih jedinica navešćemo kod objašnjenja naredbi u kojima se ove jedinice javljaju.

4.2. Složene sintaksne jedinice

Složene sintaksne jedinice služe za definisanje niski u BASIC-jeziku koje imaju određeno značenje i mogu same

za sebe stajati u programu. Ovakve niske ćemo zvati *složene konstrukcije* u jeziku. Ovo odgovara rečenicama u prirodnim jezicima, jer rečenicama izražavamo određene misli. Složene sintaksne jedinice u BASIC-jeziku su:

— naredba,
— programski red,
— potprogram i
— program.

Naredba je najprostiji oblik složene sintaksne jedinice. Prema značenju naredba može biti izvršna ili opisna. *Izvršna naredba* izražava određenu akciju koju računar treba da izvrši, a *opisna naredba* sadrži informacije koje mogu biti korisne u programu ili su neophodne za pravilno izvršavanje određenih izvršnih naredbi. Izvršne naredbe, prema vrsti akcije koju definišu, mogu biti naredbe obrade, upravljanja ili prenosa.

Programski red je niz simbola kojima se saopštava jedna ili više naredbi BASIC-jezika. Ako se na početku reda nalazi broj reda, onda se kaže da je to *označen* programski red, a ako broj reda ne postoji kaže se da je to *neoznačen* programski red (vidi odeljak 2.1).

Potprogram čini jedan ili više programskih redova. Osnovni smisao potprograma jeste izdvajanje jednog ili više programskih redova u jednu celinu, koja se može proizvoljan broj puta izvršavati u programu, bez obzira što je samo jedanput zapisan.

Program može biti sastavljen od jednog ili više programskih redova ili potprograma. Programski redovi koji čine program, odnosno potprogram, uređeni su po rastućim vrednostima brojeva redova.

Kratak izvod

- Sintaksne jedinice predstavljaju pojmove koje definišemo u BASIC-jeziku.
- Elementarne sintaksne jedinice ne mogu same za sebe stajati u programu, i to su: podatak, promenljiva, niz i izraz.
- Podatak može biti brojni ili azbučni.
- Promenljiva može biti brojna ili azbučna.
- Više brojnih ili azbučnih podataka koji imaju zajedničko ime zovu se niz.
- Izraz može biti brojni, azbučni ili relacijski.
- Složene sintaksne jedinice mogu same za sebe stajati u programu, i to su: naredba, programski red, potprogram i program.
- Naredba može biti izvršna ili opisna.
- Programski red se sastoji od jedne ili više naredbi. Programski red može biti označen ili neoznačen.
- Potprogram se sastoji od jednog ili više programskih redova, koji se mogu više puta izvršavati pri izvršavanju programa.
- Program se sastoji od jednog ili više programskih redova ili potprograma.

5
LINIJSKI PROGRAMI

Programi koji se sastoje od naredbi obrade, ulaza i izlaza zovu se *linijski programi*. Karakteristika ovakvih programa je da se sve naredbe programa izvršavaju po jedanput, počev od prve do poslednje naredbe programa, pri jednom izvršavanju programa. Drugim rečima, to su programi u kojima ne postoje upravljačke naredbe, pa se naredbe programa izvršavaju u rastućem redosledu brojeva programskih redova.

5.1. Naredba obrade brojnih podataka

Pre nego što se upoznamo sa naredbom obrade brojnih podataka, neophodno je da se upoznamo sa pravilima pisanja brojnih podataka, promenljivih i izraza. To su zapravo elementarne konstrukcije u jeziku od kojih se sastoji naredba obrade brojnih podataka.

Brojni podatak

Brojni podatak je broj u dekadnom brojnom sistemu zapisan u pozicionom ili eksponencijalnom zapisu. U *pozicionom zapisu* celobrojni i razlomljeni deo broja se razdvajaju decimalnom tačkom (na primer 12.75). Ako je celobrojni deo jednak nuli, tada ispred razlomljenog dela može stajati samo tačka (na primer: 0.34 ili .34). Ako razlomljeni deo broja ne postoji, tada iza celobrojnog dela može stajati decimalna tačka, ali ne mora (na primer: 32 ili 32.). Za pozitivne brojeve može se navesti, ispred broja, znak plus, ali ne mora, dok se za negativne brojeve mora navesti znak minus (na primer: +25 ili 25, —412.5). *Eksponencijalni zapis* se dobija kada se iza pozicionog zapisa navede slovo E, a zatim dvocifren dekadni broj, eksponent broja 10, ispred kojeg može stajati znak plus, ali ne mora, za pozitivan eksponent, i obavezno znak minus za negativan eksponent (na primer: 12.3E4 je zapis broja $12{,}3\times10^4$)

Računar ne može prihvatiti neograničeno velike brojeve. Zato moramo znati i dozvoljen *brojni interval* za brojeve koje saopštavamo računaru. Računar GALAKSIJA može da prihvati brojeve iz intervala

$$[-1.70141\times10^{38},\ 1.70141\times10^{38}]$$

Brojevi van ovog intervala ne mogu se registrovati u memoriji računara, pa ako pokušamo da ih saopštimo računaru, računar će javiti grešku. Takođe, ako se pri izvršavanju programa izračuna međurezultat ili rezultat van navedenog intervala biće dobijena poruka o otkrivenoj grešci i izvršavanje programa prekinuto.

Ostaje još jedno značajno pitanje kada je reč o brojnim podacima, a to je broj značajnih cifara. *Značajne cifre* broja su sve cifre brojnog podatka, osim nula koje služe za određivanje pozicije cifara u zapisu broja. Tako, na primer, broj 305000 ima tri značajne cifre 305, a tri nule koje slede imaju ulogu da odrede poziciju značajnih cifara. Isto tako, broj 0,001067 ima četiri značajne cifre 1067. U zapisu brojeva za računar GALAKSIJU dozvoljeno je da broj ima najviše 6 do 7 značajnih cifara. Ako korisnik zapiše veći broj značajnih cifara, tada će 6 do 7 značajnih cifara veće težine biti prihvaćeno od strane računara, a ostale cifre biće zamenjene nulama ili odbačene. Tako, broj 987654321 biće u računar unet kao broj 987654000. Dakle, tri značajne cifre manje težine su zamenjene nulama.

Oba navedena ograničenja za brojne podatke, i u pogledu brojnog intervala i broja značajnih cifara, potiču od načina registrovanja brojeva u memoriji računara. Bez obzira na način pisanja brojeva, oni se u memoriji računara registruju u obliku pokretnog zareza [2].

Primer 1. Sledeći brojevi su u pozicionom zapisu:
15 —142 305 —.102 14.75 +18

Primer 2. Sledeći brojevi su u eksponencijalnom zapisu:
25E—4 predstavlja broj 25×10^{-4}
—.42E12 predstavlja broj -0.42×10^{12}
12E4 predstavlja broj 12×10^{4}
—14.75E—1 predstavlja broj -14.75×10^{-1}

Primer 3. Sledeće brojeve računar ne može prihvatiti, jer su van dozvoljenog brojnog intervala:
1.25E39 —10.5E38 1E40 —0.5E40

Primer 4. Broj 23.12345 sadrži 7 značajnih cifara, pa će značajna cifra najmanje težine biti odbačena pri registrovanju broja u memoriji računara. Tako će ovaj broj biti registrovan kao 23.1234.

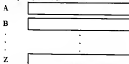
Brojna promenljiva

Brojna promenljiva je simbolička oznaka kojoj se može dodeliti brojni podatak. Svaku promenljivu odlikuje ime promenljive, oblast definisanosti i tekuća vrednost. *Ime brojne promenljive* može biti slovo latinske azbuke. Kako latinska azbuka sadrži 26 slova to se može definisati 26 imena promenljivih. Tako su imena promenljivih A, B, X, Y itd. Skup brojnih podataka od kojih svaki može biti dodeljen kao vrednost promenljive zove se *oblast definisati* promenljive. U našem slučaju oblast definisanosti brojnih promenljivih je brojni interval brojnih podataka, tj.

$$[-1.70141 \times 10^{38}, 1.70141 \times 10^{38}]$$

Broj, iz oblasti definisanosti, dodeljen promenljivoj zove se *tekuća vrednost* promenljive.

Prostor u memoriji računara rezervisan za čuvanje tekućih vrednosti promenljivih zove se *zona promenljivih*. Kako postoji 26 brojnih promenljivih, to se zona promenljivih sastoji od 26 registara, pri čemu svaki registar sadrži 32 ćelije (4 bajta). Ime promenljive se javlja kao simbolička adresa registra a sadržaj registra kao tekuća vrednost promenljive (sl. 5.1.1).



Sl 5.1.1 Zona brojnih promenljivih

Primer 1. Sledeća slova su imena promenljivih
A C T X Y Z

Primer 2. Sledeće niske nisu imena promenljivih:
- AB — dva slova ne mogu da obrazuju ime promenljive,
- X1 — slovo i cifra ne mogu da obrazuju ime promenljive,
- Ž — slovo Ž jugoslovenske azbuke ne može da obrazuje ime promenljive.

Brojni izraz

Brojni izraz se piše na način koji je uobičajen u matematici, s tim što se mogu koristi samo otvorene i zatvorene male zagrade. Argumenti brojnog izraza mogu biti: brojni podatak, brojna promenljiva i numerička funkcija. Brojni podatak i brojnu promenljivu smo već upoznali, a numeričke funkcije ćemo upoznati kasnije. Nad argumentima brojnog izraza definisane su operacije sabiranja, oduzimanja, množenja i deljenja. Ovo su operacije nad dva argumenta. Pored ovih operacija postoji i operacija promene znaka koja je definisana nad jednim argumentom. U tabeli 5.1.1 prikazani su prioriteti pojedinih aritmetičkih operacija. Kao što se vidi promena znaka ima najviši prioritet, zatim deljenje i množenje i na kraju najniži prioritet imaju

Operacija	Simbol	Prioritet
Promena znaka	—	1
Deljenje	/	2
Množenje	×	2
Oduzimanje	—	3
Sabiranje	+	3

Tabela 5.1.1. Aritmetičke operacije

oduzimanje i sabiranje. Kada brojni izraz sadrži više od jedne operacije, onda se vrednost izraza određuje tako što se operacije višeg ili istog prioriteta izvršavaju sleva nadesno. Kada se želi promeniti propisan prioritet, tada se koriste zagrade. Deo brojnog izraza zapisan između zagrada dobija najviši prioritet. Ako postoji veći broj zagrada, tada su unutrašnje zagrade višeg prioriteta od spoljašnjih. Izračunavanjem vrednosti brojnog izraza dobija se jedan rezultujući brojni podatak.

Aritmetička operacija deljenja može na računaru da proizvede teškoće ako je delilac jednak nuli. U ovom slučaju važi pravilo: Ako su deljenik i delilac jednaki nuli, tada će rezultat biti nula. Ako je deljenik različit od nule, a delilac jednak nuli tada će rezultat biti nedefinisan i računar će prekinuti rad i izdati poruku HOW?

Primer. U tabeli 5.1.2 dati su primeri zapisa brojnih izraza u matematici i u BASIC-jeziku

Brojni izraz	
u matematici	u BASIC-jeziku
5,28	5.28
x	X
$a \cdot b - c \cdot d$	A*B—C*D
$-a \cdot \frac{b}{c+d}$	—A*B/(C+D)
$\frac{b+c}{d+5}$	(B+C)/(D+5)
$a+\frac{b}{c}+d$	A+B/C+D

Tabela 5.1.2. Primeri brojnih izraza

Aritmetička naredba

Naredba kojom se izračunata vrednost brojnog izraza može dodeliti brojnoj promenljivoj zove se a r i t m e t i č k a   n a r e d b a ili naredba obrade brojnih podataka. Ova naredba je definisana sa:

<brojna promenljiva> = <brojni izraz>

Značenje ove naredbe je sledeće: izračunava se vrednost brojnog izraza, a zatim izračunata vrednost dodeljuje promenljivoj na levoj strani znaka jednakosti. Promenljivim koje se javljaju kao argumenti u brojnom izrazu moraju biti dodeljene vrednosti pre izvršavanja aritmetičke naredbe. Ovde je važno uočiti razliku u značenju znaka jednakosti u aritmetičkoj naredbi i uobičajenog značenja u matematici. Znak jednakosti u aritmetičkoj naredbi simbolizuje proces koji se sastoji od sledeća dva koraka:

— izračunavanja vrednosti brojnog izraza i
— dodeljivanja izračunate vrednosti brojnoj promenljivoj na levoj strani znaka jednakosti.

Kako se navedeni koraci izvršavaju jedan za drugim, to je moguće da se u okviru brojnog izraza, nalazi i ime promenljive sa leve strane znaka jednakosti. Tekuća vrednost ove promenljive koristi se u prvom koraku za izračunavanje vrednosti brojnog izraza, a zatim u drugom koraku izračunata vrednost se dodeljuje ovoj, istoj, promenljivoj kao nova tekuća vrednost.

Primer 1. Aritmetička naredba

P=3.14159

dodeljuje konstantu 3.14159 promenljivoj P

Primer 2. Aritmetička naredba

N=N+1

uvećava tekuću vrednost promenljive N za 1 i tako dobijenu vrednost dodeljuje kao novu tekuću vrednost promenljivoj N.

5.2. Naredba obrade azbučnih podataka

U mnogim zadacima, koje rešavamo pomoću računara, pored brojnih podatak javljaju se i podaci u vidu teksta, kao što su nazivi gradova, vrsta zanimanja i sl. U ovom odeljku ćemo definisati ovakvu vrstu nenumeričkih podataka.

Azbučni podatak

Niska osnovnih simbola zapisana između znakova navoda zove se *azbučni podatak.* Broj osnovnih simbola, između znakova navoda, zove se *dužina azbučnog podatka.* Ako se između znakova navoda ne navede nijedan simbol, onda se kaže da je to *prazan azbučni podatak.* Dužina praznog podatka je nula

Primer. Sledeće niske simbola su azbučni podaci:
"BEOGRAD"
"15 AVGUST 1979"
"X—Y"
" "
""

Poslednji azbučni podatak je prazan. Međutim, azbučni podatak " " ima dužinu 1, jer sadrži jedan osnovni simbol — prazninu (blanko).

Azbučna promenljiva

Promenljiva kojoj se kao vrednost može pridružiti azbučni podatak zove se *azbučna promenljiva.* U BASIC-jeziku računara GALAKSIJA moguće je koristiti samo dve azbučne promenljive čija su imena X $ i Y $. Po prisustvu znaka

za dolar ($) razlikuje se ime azbučne promenljive od imena brojne promenljive. Prostor u memoriji koji se rezerviše za čuvanje tekućih vrednosti azbučnih promenljivih zove se *zona azbučnih promenljivih* (sl. 5.2.1). Maksimalna dužina azbučnog podatka koji se može dodeliti azbučnoj promenljivoj iznosi 16. Prema tome, kako postoje dve azbučne promenljive (X $ i Y $), to će zona azbučnih promenljivih zahtevati 2×16=32 bajta u memoriji.

X $ [ ]

Y $ [ ]

Sl. 5.2.1. Zona azbučnih promenljivih

Azbučni izraz

Mi ćemo uvesti pojam azbučnog izraza po analogiji sa brojnim izrazom. *Azbučni izraz* predstavlja zapis postupka za dobijanje jednog rezultirajućeg azbučnog podatka, na osnovu jednog ili više azbučnih podataka. Dobijeni rezultirajući azbučni podatak zove se *vrednost azbučnog izraza*. Argument azbučnog izraza može biti azbučni podatak, azbučna promenljiva i azbučna funkcija. Između argumenata azbučnog izraza nalazi se znak operacije dopisivanja (konkatenacije) u oznaci +. Operacija dopisivanja se izvodi tako, što se na vrednost argumenta levo od znaka dopisivanja dopiše zdesna vrednost argumenta desno od znaka dopisivanja. Operacije dopisivanja u azbučnom izrazu se izvode sleva nadesno.

Primer

Azbučni izraz

X $+Y $+ "—1984"

ima vrednost

1—MAJ—1984

ako azbučne promenljive X $ i Y $ imaju redom vrednosti "1—" i "MAJ".

Naredba obrade azbučnih podataka

Naredba obrade azbučnih podataka slična je naredbi obrade brojnih podataka, sa razlikom što se sada u naredbi javljaju azbučne veličine. Naredba se piše u obliku:

⟨azbučna promenljiva⟩=⟨azbučni izraz⟩

Značenje ove naredbe je sledeće: određuje se vrednost azbučnog izraza i ovako određen azbučni podatak se dodeljuje azbučnoj promenljivoj sa leve strane znaka jednakosti. Ako je azbučni podatak određen kao vrednost azbučnog izraza veće dužine od 16, tada će se javiti greška WHAT? i

prekinuti dalje izvršavanje programa. Postoji jedno važno ograničenje: ako se azbučna promenljiva sa leve strane znaka jednakosti javlja u azbučnom izrazu na desnoj strani znaka jednakosti, tada se ova promenljiva mora javiti kao prva sleva u azbučnom izrazu. Tako, naredba

X $=X $+"BEOGRAD"

je korektno zapisana, dok zapis X $="BEOGRAD"+X $ nije korektan!

Primer 1.

Naredba obrade azbučnih podataka

X $="BEOGRAD"

dodeljuje promenljivoj X $ azbučni podatak "BEOGRAD".

Primer 2.

Naredba obrade azbučnih podataka

Y $=X $

dodeljuje tekuću vrednost azbučne promenljive X $ promenljivoj Y $.

Primer 3.

Naredba

X $=" "

dodeljuje azbučnoj promenljivoj X$ prazan azbučni podatak. Ovo je različito od naredbe X$="␣" koja dodeljuje promenljivoj azbučni podatak koji se sastoji od jednog znaka blanko.

5.3. Naredba izlaza

Naredbe obrade omogućuju različita izračunavanja. Rezultati takvih izračunavanja nalaze se u memoriji računara, kao tekuće vrednosti promenljivih. Da bi čovek koristio ove rezultate, oni moraju biti iz memorije preneti i registrovani na medijum sa kojeg će ih čovek lako koristiti. Takav medijum je papir ili ekran. Prostor na papiru ili ekranu u kome se može registrovati jedan simbol zove se *ćelija*. Niz horizontalno raspoređenih ćelija čine *red*, a više redova obrazuju *stranu*. Jedna ili više strana obrazuju *izlazni izveštaj*. Prema tome, izlazni izveštaj sadrži sve rezultate rada jednog programa. U mnogim slučajevima vrlo je važno da ovaj dokument bude što pregledniji, jer ga često koriste ljudi koji nisu upućeni u računarsku tehniku i programiranje. Uređenje izlaznog izveštaja postiže se pogodnim rasporedom simbola na papiru ili ekranu. Može se zamisliti da su ćelije u redu numerisane sleva nadesno sa 0, 1, 2, . . . Ovaj redni broj ćelije zove se *pozicija ćelije*, a niz ćelija u kojima se registruje jedan podatak zove se *polje*.

Registrovanje podataka na papiru ili ekranu vrši se tako, što se podaci šalju iz memorije i registruju u tekućem redu. Svaki prelazak na sledeći red onemogućuje povratak u prethodni red. Prema tome, redovi se obrazuju sekvencijalno jedan po jedan. Naredba kojom se podaci

šalju iz memorije na izlazni organ zove se *naredba izlaza*. Niska simbola koja se prenosi iz memorije na izlazni organ, zove se *izlazni slog*.

Broj ćelija u jednom redu izlaznog izveštaja zove se *dužina reda* i može biti različita kod raznih izlaznih uređaja. Tako, ekran računara GALAKSIJA sadrži 32 ćelije u jednom redu, a jednu stranu čine 16 redova koliko se jednovremeno može prikazati na ekranu. Sintaksa naredbe izlaza može se opisati na sledeći način:

$$\text{PRINT}\left[\left[\left\{\begin{matrix}\langle\text{izraz}\rangle\\ \text{AT}\,\langle\text{brojni izraz}\rangle\end{matrix}\right\}\left\{\begin{matrix};\\ ,\end{matrix}\right\}\right]^{*}\left[\left\{\begin{matrix}\langle\text{izraz}\rangle\\ \text{AT}\,\langle\text{brojni izraz}\rangle\end{matrix}\right\}\left[\begin{matrix};\\ ,\end{matrix}\right]\right]\right]$$

gde izraz može biti brojni ili azbučni izraz. Ovde postoji ograničenje za oblik azbučnog izraza, tako da se može javiti samo jedan azbučni podatak, jedna azbučna promenljiva ili funkcija CHR$ (vidi odeljak 5.7). Elemente iza službene reči PRINT zovemo *lista naredbe izlaza*. Naredba izlaza ima sledeće značenje:

— od vrednosti brojnih i azbučnih izraza obrazuje se izlazni slog,
— simboli: zarez (,) i tačka-zarez (;) služe za razdvajanje elemenata u listi naredbe izlaza i za uređenje izlaznog sloga,
— ovako uređen izlazni slog izdaje se na ekranu sleva nadesno i može zauzeti jedan ili više redova na ekranu,

j \ i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0																																
32																																
64																																
96																																
128																																
160																																
192																																
224																																
256																																
288																																
320																																
352																																
384																																
416																																
448																																
480																																

Sl. 5.3.1. Numeracija pozicija na ekranu za AT opciju

— ako se navede samo službena reč PRINT bez liste, tada se izdaje jedan prazan red na ekranu,
— ako se koristi AT (brojni izraz), tada celobrojna vrednost brojnog izraza određuje početak izlaza dela sloga koji sledi na ekranu, pri čemu su sve pozicije na ekranu numerisane od 0 do 511 (sl. 5.3.1).

Na slici se pozicija na ekranu dobija kao zbir i+j. Uređenje izlaznog sloga pomoću AT-opcije naročito je pogodno za definisanje početka izlaznog sloga. Međutim, može se ova opcija javiti više puta u listi naredbe PRINT i na taj način definisati izlaze delova izlaznog sloga po celom ekranu. Da bismo mogli da planiramo izgled izlaznog izveštaja moramo znati način izdavanja brojnih i azbučnih podataka, kao i mogućnosti uređenja izlaznog sloga. Za izdavanje brojnog podatka važi sledeće:

1. Ako brojni podatak, koji se izdaje, po apsolutnoj vrednosti, pripada intervalu [0.1,999999], tada se izdaje u pozicionom obliku, s tim što se na poziciji za znak izdaje praznina (blanko) u slučaju pozitivnog broja, odnosno znak — za negativan broj.
2. Ako brojni podatak, koji se izdaje, po apsolutnoj vrednosti, ne pripada intervalu [0.1,999999], tada se izdaje u eksponencijalnom obliku, pri čemu se u celobrojnom delu mantise broja nalazi jedna značajna cifra.
3. Brojni podaci se izdaju sa najviše 6 značajnih cifara.

Azbučni podatak se izdaje u istom obliku kako je i naveden u listi naredbe izlaza, s tim što se znaci navoda koji ga omeđavaju ne izdaju.

Simboli za uređenje izlaznog izveštaja unutar jednog reda imaju sledeće značenje:

1. Simbol tačka-zarez (;) služi za razdvajanje izraza u listi naredbe izlaza, ako se ovi ne razdvajaju zarezom. Ako se ovaj simbol nađe na kraju liste naredbe izlaza, to znači da izlazni uređaj ostaje u tekućem redu i ne prelazi u sledeći red. Ovaj simbol nema smisla pisati na početku liste naredbe izlaza, a takođe nema smisla navoditi više ovih simbola jedan iza drugog.

2. Simbol zarez (,) definiše prelazak na početak sledećeg polja od 8 ćelija jednog reda na ekranu. Dakle, u ovom slučaju se uzima da red na ekranu sadrži 4 polja od po 8 ćelija, i to tako da je (sl. 5.3.1):
— od 0 do 7. pozicije 1. polje,
— od 8. do 15. pozicije 2. polje,
— od 16. do 23. pozicije 3. polje,
— od 24. do 31. pozicije 4. polje.

Ovaj simbol se ne može javiti pre prvog podatka u listi naredbe. Takođe se ne može navesti više zareza jedan pored drugog.

Primer 1.

Naredba izlaza
PRINT — 35.2,0.0015
izdaje izlazni slog u obliku
— 35.2 1.5E — 03

Ovaj primer pokazuje:

— da se brojevi po apsolutnoj vrednosti iz intervala [0.1;999999] izdaju u pozicionom obliku,
— da se brojevi koji ne pripadaju navedenom intervalu izdaju u eksponencijalnom obliku,
— da se znak broja izdaje ako je broj negativan (—), a da se za pozitivne brojeve na poziciji za znak izdaje praznina,
— tačka-zarez razdvaja elemente u listi naredbe izlaza

Primer 2.

Naredba izlaza

PRINT 3;" JE MANJE OD ";5

izdaje izlazni slog

3 JE MANJE OD 5

Ovaj primer pokazuje da se azbučni podatak izdaje u navedenom obliku, pri čemu se znaci navoda ne izdaju

Primer 3.

Ako naredbe izlaza

10 PRINT "TRIGLAV JE ␣";

20 PRINT "VISOK 2864 METARA."

čine program, onda se izvršavanjem ovog programa primenom komande RUN, izdaje tekst u obliku

TRIGLAV JE VISOK 2864 METARA.

Ovaj primer pokazuje da tačka-zarez na kraju liste naredbe izlaza ostavlja izlazni uređaj u tekućem redu.

Primer 4.

Naredba izlaza

PRINT 9.876543, —2.45,3

izdaje izlazni slog u obliku:

red na ekranu	9.87654	—2.45	3	
pozicija	0	8	16	
polje u redu	1	2	3	4.

Ovaj primer pokazuje:

— brojni podatak se izdaje sa najviše 6 značajnih cifara
— zarez između elemenata liste razdvaja elemente, ali i definiše izdavanje sledećeg elementa u sledećem polju od 8 ćelija na ekranu. Tako je broj —2.45 izdat u drugom polju, a broj 3 u trećem polju u redu na ekranu.

Primer 5.

Naredba obrade i naredba izlaza daju nam mogućnost da vršimo različita izračunavanja i rezultate izdajemo na ekranu. Ove naredbe možemo izvršavati u ne-

posrednom režimu i na taj način raditi sa računarom slično kao što se radi sa elektronskim kalkulatorom. Razlika će biti u tome što se kod kalkulatora određena operacija izvodi na pritisak odgovarajućeg tastera, a ovde se preko tastature saopštava odgovarajuća naredba BASIC-jezika.

Pokažimo ovo na primeru. Neka treba izračunati vrednost sledećeg brojnog izraza:

$$\frac{2,52+4,55\times3,12}{15,12-2,55}$$

Ovo se može uraditi neposrednim izvršavanjem naredbe PRINT:

```
>PRINT (2.52+4.55*3.12)/(15.12—2.55)
 1.32983
```

Korisnik može zahtevati izvršavanje neke naredbe u neposrednom režimu samo ako na ekranu postoji poruka interpretatora (>) da se očekuje saopštenje korisnika. U ovom primeru, iza poruke interpretatora (>) uneta je naredba PRINT bez broja reda, što znači da će se naredba izvršiti u neposrednom režimu, a u listi ove naredbe naveden je izraz čiju vrednost želimo odrediti. Pritiskom na taster ENTER vrši se unošenje gornje naredbe i njeno izvršavanje, koje se sastoji u izračunavanju vrednosti navedenog izraza i izdavanju izračunate vrednosti (1.32983).

Primer 6.

Možemo koristiti i imena promenljivih za čuvanje brojnih podataka, da ne bi svaki put unosili brojne vrednosti kada se više puta računa po istoj formuli.

Neka treba računati površinu kruga za razne vrednosti poluprečnika. Tako, ako treba izračunati površinu kruga za poluprečnike $r_1=2,5$ cm i $r_2=3,62$ cm, ovo se može uraditi na sledeći način:

```
>P=3.14159
>P.2.5*2.5*P
 19.6349
>P.3.62*3.62*P
 41.1686
```

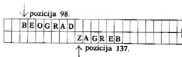
Neposrednim izvršavanjem naredbe P=3.14159, vrednost broja je dodeljena promenljivoj P, a zatim neposrednim izvršavanjem odgovarajućih PRINT naredbi izvršena su tražena izračunavanja. Ovde smo primenili skraćeno pisanje službene reči PRINT, tako da je navedeno samo prvo slovo reči iza kojeg sledi tačka (P.). Ovo je naravno praktično koristiti kada računar koristimo kao kalkulator.

Primer 7.

Naredba

```
PRINT AT98, "BEOGRAD", AT137, "ZAGREB"
```

izdaje tekst BEOGRAD i ZAGREB na sledeći način:



5.4 Kraj programa

Postoje dve vrste kraja programa. Može se govoriti o kraju programa u smislu poslednje naredbe koja se izvršava i sa kojom se prekida rad programa, a može biti reč o fizičkom kraju programa.

Kraj izvršavanja programa

Ovom naredbom se završava rad po programu. Ova naredba se postavlja na onim mestima programa koja odgovaraju kraju algoritma u algoritamskim šemama. Kako jedan algoritam može imati više završetaka, tako se i više ovih naredbi može nalaziti u programu. Naredba kraja izvršavanja programa se piše u vidu jedne službene reči

STOP

Kada se pri izvršavanju programa dođe do ove naredbe, završava se rad po programu i izdaje poruka

READY
> _

Fizički kraj programa

Ne postoji naredba kojom se raspoznaje fizički kraj programa. Program se fizički završava programskim redom koji ima najveći broj reda, bez obzira na naredbe koje se nalaze u ovom programskom redu. Ako se u poslednjem programskom redu nalazi naredba koja se ne prenosi upravljanje na neku od naredbi programa, tada se izvršavanjem poslednje naredbe u programu završava izvršavanje programa, kao da je sledeća naredba STOP. Na primer, ako je poslednja naredba

PRINT A, B

tada će biti izdate vrednosti promenljivih A i B i završiće se izvršavanje programa. Dobra programerska praksa je da poslednja naredba u programu bude naredba STOP.

5.5. Komentari u programu

U programu je često korisno pisati komentare koji objašnjavaju program ili pojedine delove programa. Ovo je omogućeno naredbom za pisanje komentara koja ima oblik

![⟨osnovni simbol⟩]*

Ovo je opisna naredba koja se može nalaziti bilo gde u programu. Dolazak na ovu naredbu za vreme izvršavanja programa ne proizvodi nikakvo dejstvo i prelazi se na sledeću naredbu programa.

Primer.

Napisati program koji izračunava vrednost polinoma:

$$P(x)=x^3-2{,}5x^2+4{,}2$$

za $x=1{,}5$. Program se može zapisati u obliku:

```
10 !VREDNOST POLINOMA
20 X=1.5
30 P=X*X*X—2.5*X*X+4.2
40 PRINT "ZA X=";X;" P(X)=";P
50 STOP
```

U redu broj 10 nalazi se naredba za komentar (!) u kojoj je naveden komentar VREDNOST POLINOMA koji objašnjava funkciju programa. Ovim primerom želimo da objasnimo ceo postupak izvršavanja jednog programa na računaru. Za ovaj primer možemo dobiti sledeći protokol na ekranu o unošenju i izvršavanju programa:

```
> NEW
> 10 !VREDNOST POLINOMA
> 20 X=1.5
> 30 P=X*X*X—2.5*X*X+4.2
> 40 PRINT "ZA X=";X;" P(X)=";P
> 50 STOP
> RUN
ZA X=1.5 P(X)=1.95
> _
```

Komandom NEW izvršeno je brisanje zone programa, a zatim su unete naredbe programa sa brojevima redova 10, 20, 30, 40 i 50. Ovako unet program se nalazi u zoni programa. Komandom RUN zadaje se izvršavanje programa. Prva naredba programa (red 10) je komentar i nema nikakvog efekta pri izvršavanju programa. Druga naredba vrši dodeljivanje vrednosti 1,5 promenljivoj X, a u trećoj se vrši izračunavanje vrednosti polinoma i izračunata vrednost dodeljuje se promenljivoj P. Naredba PRINT vrši izdavanje rezultata u obliku

ZA X=1.5 P(X)=1.95

posle čega se program završava naredbom STOP i na ekranu se izdaje poruka (>) korisniku za unošenje nove komande.

5.6. Naredba ulaza

Programi u kojima postoje samo naredbe obrade i naredbe izlaza daju pri svakom izvršavanju iste rezultate. Prema tome, ovakve programe ima smisla izvršiti samo jedanput. Međutim, programi koji sadrže promenljive koje dobijaju vrednosti preko ulaznog organa, pri izvršavanju programa, daju rezultate koji zavise od ulaznih veličina. Ovo je i najvažnije obeležje programiranja. Zapravo, racionalno korišćenje računara u raznim primenama sastoji se u eksploataciji programa, a to znači u izvršavanju programa za različite ulazne podatke. Zato programe treba pisati tako, da služe za rešavanje određene vrste zadataka, a ne jednog konkretnog zadatka. Tako, ako želimo da rešimo kvadratnu jednačinu, napisaćemo program za rešavanje bilo koje kvadratne jednačine, oblika

$$ax^2+bx+c=0$$

pri čemu ćemo koeficijente a, b i c uzeti kao ulazne veličine. Pogrešno bi bilo pisati program za rešavanje konkretne kvadratne jednačine, recimo

$$x^2+4x-2=0$$

Pisanjem programa za opšti oblik kvadratne jednačine mi ćemo raspolagati programom za rešavanje kvadratne jednačine, a u konkretnom slučaju izvršićemo program za odgovarajuće ulazne veličine. U navedenom primeru ulazne veličine biće a=1, b=4, c=−2.

Naredba kojom se dodeljuje vrednost promenljivoj preko ulaznog organa računara zove se *naredba ulaza*. Ova naredba prenosi podatak sa ulaznog organa u zonu promenljivih u RAM-memoriji računara. Naredba ulaza ima oblik

INPUT ⟨promenljiva⟩

U našem slučaju ulazni organ je tastatura, tako da naredba ulaza dodeljuje podatak unet preko tastature promenljivoj navedenoj iza službene reči INPUT. Promenljiva može biti brojna ili azbučna. Ako je u naredbi ulaza navedena brojna promenljiva, tada i podatak koji se unosi mora biti brojni podatak. Međutim, ako je promenljiva azbučna, onda i podatak koji se unosi mora biti azbučni. Azbučni podatak se može unositi između znakova navoda, ali i bez znakova navoda. Dužina azbučnog podatka ne sme biti veća od 16. Ako je dužina azbučnog podatka veća od 16, tada će se uneti prvih 16 znakova sleva, a ostali znaci će biti odbačeni.

Primer 1.

Izračunavanje vrednosti polinoma

$$P(x)=x^3-2{,}5x^2+4{,}2$$

sada možemo sastaviti tako da se izračunavanje vrši za bilo koju vrednost *x* saopštenu na ulazu računara, preko tastature. Tako, program može imati sledeći izgled

```
10 !VREDNOST POLINOMA
20 INPUT X
30 P=X*X*X—2.5*X*X+4.2
40 PRINT "ZA X=";X;" P(X)=",P
50 STOP
```

Na uobičajeni način možemo uneti ovaj program u zonu programa. Izvršavanjem programa dobijamo sledeći protokol na ekranu:

```
> RUN
?1.5
ZA X=1.5 P(X)=1.95
> _
```

Kada se pri izvršavanju programa dođe do naredbe ulaza, računar izdaje znak pitanja (?) i očekuje da korisnik unese odgovarajući podatak. Kada korisnik unese podatak i pritisne na taster ENTER, tada se izvrši dodeljivanje unetog podatka promenljivoj u naredbi ulaza i nastavi izvršavanje programa.

Primer 2.

Možemo prethodni program sastaviti tako da korisnik odgovara na pitanje pri saopštavanju ulaznih veličina. Ovo se vrlo često primenjuje kod programa pisanih za interaktivno korišćenje. Program možemo zapisati u obliku:

```
10 !VREDNOST POLINOMA
20 PRINT "VREDNOST ARGUMENTA",
30 INPUT X
40 P=X*X*X—2.5*X*X+4.2
50 PRINT "VREDNOST POLINOMA JE",P
60 STOP
```

Izvršavanjem ovog programa dobija se sledeći protokol:

```
> RUN
VREDNOST ARGUMENTA?1.5
VREDNOST POLINOMA JE 1.95
> _
```

Kao što se vidi program izdaje tekst VREDNOST ARGUMENTA? i zatim računar očekuje da korisnik unese brojni podatak. Kada je saopšten brojni podatak računar nastavlja rad. Na ovaj način tekst koji se izda pre unošenja podatka podseća korisnika kakav podatak treba da unese. Ovo znatno olakšava korišćenje programa od strane korisnika, koji često nisu autori programa.

5.7. Numeričke i azbučne funkcije

Za funkciju kažemo da je *numerička* ako je vrednost funkcije brojni podatak. Ove funkcije se mogu koristiti kao argumenti brojnih izraza. U BASIC-jeziku za računar GA-

LAKSIJA mogu se koristiti sledeće numeričke funkcije:

1. *Celobrojna funkcija.* To je funkcija čiji argument može biti brojni izraz, a vrednost funkcije je najveći ceo broj jednak ili manji od vrednosti argumenta funkcije. Ova funkcija se piše u obliku:

INT (b)

gde je b brojni izraz. Tako, na primer, vrednost funkcije INT(7.99) je 7, a vrednost funkcije INT(—0.12) je —1.

2. *Generisanje pseudo-slučajnih brojeva.* Ova funkcija nema argument već samo propisano ime RND. Funkcija generiše pseudo-slučajni broj iz intervala (0;1). Ova funkcija generiše pseudo-slučajne brojeve sa različitim početkom pri svakom uključivanju računara. Ovo znači da se pri svakom izvršavanju programa koji sadrži funkciju RND dobijaju različiti pseudo-slučajni brojevi. Ako se žele izračunati pseudo-slučajni brojevi iz intervala koji nije (0,1), tada se to može učiniti izračunavanjem

A+(B—A)*RND

pa će interval generisanih pseudo-slučajnih brojeva biti (A,B). Tako se izvršavanjem naredbe

PRINT, RND, RND, RND, RND

mogu dobiti sledeći pseudo-slučajni brojevi:

.989317 .793533 .924699 .683954

3. *Adresa promenljive.* To je funkcija čiji je argument promenljiva, a vrednost adresa bajta najmanje težine u memoriji u kojem se nalazi vrednost promenljive. Funkcija se piše u obliku:

PTR p

gde je p promenljiva. Tako na primer, funkcija PTR X$ ima vrednost 10864, a to je adresa polja u memoriji u kojem se nalazi tekuća vrednost azbučne pomenljive X$.

Ova funkcija se može pisati i bez argumenta. U ovom slučaju vrednost funkcije je adresa bajta u memoriji koji sledi iza slova R u nazivu funkcije (PTR). Tako, program

```
10 A=PTR
20 PRINT A
```

izdaje vrednost 11329 jer je to adresa bajta u memoriji koji sledi iza imena funkcije PTR. Ovo se može iskoristiti za dobijanje adrese određenog dela programa, na primer kod programske modifikacije (vidi primer 11.7.2).

4. *Konverziona funkcija.* To je funkcija čiji argument je adresa polja u memoriji u kojem se nalazi pozicioni ili eksponencijalni zapis brojne vrednosti ili brojnogo izraza, a vrednost funkcije je vrednost argumenta u internom kôdu računara (u pokretnom zarezu). Funkcija se piše u obliku:

VAL(a)

gde je a adresa polja u memoriji. Namesto a može stajati i brojni izraz, čija vrednost se uzima kao argument. Tako, na primer, program

```
10 INPUT X$
20 PRINT X$, VAL(PTR X$)
30 STOP
```

omogućuje unošenje azbučnog podatka i njegovo izdavanje. Ako kao azbučni podatak unesemo pozicioni ili eksponencijalni zapis broja, tada će se izvršavanjem programa dobiti sledeći izveštaj:

```
>RUN
?123.4E—3
123.4E—3 .1234
```

Kao što se vidi, unet je broj u eksponencijalnom obliku (123.4E—3) koji je izdat u istom obliku, ali i u obliku koji se dobija iz internog kôda računara (.1234).

* * *

Za funkciju kažemo da je *azbučna* ako je vrednost funkcije azbučni podatak. Ova funkcija se može koristiti kao argument azbučnih izraza. U BASIC-jeziku za računar GALAKSIJA može se koristiti azbučna funkcija:

CHR$(b)

gde je b brojni izraz. Celobrojna vrednost brojnog izraza je argument, a vrednost funkcije je znak u ASCII-kôdu koji odgovara vrednosti argumenta. U Tabeli 5.7.1 prikazane su

d_1 \ d_2	32	48	64	80
0	␣	0	⩽	P
1	!	1	A	Q
2	"	2	B	R
3	#	3	C	S
4	$	4	D	T
5	%	5	E	U
6	&	6	F	V
7	'	7	G	W
8	(	8	H	X
9	)	9	I	Y
10	*	:	J	Z
11	+	;	K	Č
12	,	<	L	Ć
13	—	=	M	Ž
14	.	>	N	Š
15	/	?	O	—

Tabela 5.7.1. Dekadne vrednosti ASCII-kôdova

dekadne vrednosti pojedinih znakova u ASCII-kôdu. Iz tabele se dekadna vrednost kôda, za odgovarajući znak, dobija kao zbir vrednosti d_1 i d_2. Tako, slovo A ima dekadnu vrednost 65, jer je $d_1+d_2=1+64=65$. Kao ilustraciju primene ove funkcije navodimo da izvršavanje naredbe

PRINT CHR$(66);CHR$(79);CHR$(82)

izdaje reč BOR

5.8. Rešeni zadaci

5.8.1. Koristeći računar kao kalkulator izračunati vrednost kamate od 7,5% i od 28% na sumu od 15000 din.
Rešenje: Sumu ćemo, kao brojni podatak, uneti u memoriju, tako što ćemo je dodeliti promenljivoj S, a zatim dva puta izvršiti naredbu PRINT za izračunavanje kamata. Protokol ima sledeći izgled:

```
> S=1500
> P.7.5*S/100
 112.5
> P.28*S/100
 420
```

Ovde je primenjeno skraćivanje službene reči PRINT u oblik P. Naravno ovo nije uticalo na navedeno izračunavanje. Rezultati su isti kao da je navedena i službena reč u obliku PRINT, ali je ovo brže sa gledišta unošenja naredbe.

5.8.2. Zadatak 5.8.1. možemo uopštiti i rešiti u obliku programa: Sastaviti program koji za unetu sumu izračunava iznos kamate, pri čemu se kamata unosi u procentima sa ulaza.
Rešenje: Program se sastoji od naredbi kojima se unosi suma i kamata u procentima, zatim se izračunava vrednost kamate i na kraju izdaje izračunata kamata. U ovom slučaju protokol ima sledeći izgled:

```
> NEW
> 10 INPUT S
> 20 INPUT P
> 30 K=S*P/100
> 40 PRINT K
> 50 STOP
> RUN
?1500
?7.5
 112.5
> RUN
?1500
?28
 420
```

Komandom NEW je izbrisan ranji program u zoni programa, zatim je unet program od 10. do 50. reda. Izvršava-

nje programa je zadato komandom RUN, a zatim uneti brojevi 1500 i 7.5, a program je izračunao vrednost kamate i izdao broj 112.5 Na sličan način je ponovljeno izvršavanje programa (nova komanda RUN).

5.8.3. Zadatak 5.8.2 možemo rešiti na mnogo načina. U ovom primeru želimo da prikažemo rešenje koje će biti udobnije sa gledišta korišćenja, tako što će izveštaj na ekranu biti čitljiv.

Rešenje: Interaktivno korišćenje računara podrazumeva saopštavanje čitljivih poruka od strane računara, a takođe i lako unošenje podataka od strane korisnika programa. Poštujući ove zahteve program se može napisati u obliku

```
10 PRINT "UNESITE SUMU";
20 INPUT S
30 PRINT "UNESITE PROCENAT",
40 INPUT P
50 K=S*P/100
60 PRINT "VREDNOST KAMATE:",K
70 STOP
```

Izvršavanjem ovog programa dobija se sledeći protokol:

```
> RUN
UNESITE SUMU?1500
UNESITE PROCENAT?7.5
VREDNOST KAMATE: 112.5
> _
```

Kao što se vidi korisnik dobija od računara poruke, a ne samo znak pitanja, što mu omogućuje da prave podatke unosi u pravo vreme. Ako računar izdaje samo znake pitanja korisnik bi mogao lako da pogrešno unese podatke. Na primer, da zameni sumu i procenat, pa da prvo unese procenat a zatim sumu, što bi, u nekim primerima, moglo dovesti do pogrešnog rezultata.

Kratak izvod

- **Brojni podatak**
  - — može se pisati u pozicionom ili eksponencijalnom obliku,
  - — brojni interval $[-10^{38}; 10^{38}]$,
  - — 6 do 7 značajnih cifara
- **Brojna promenljiva** može biti slovo latinske azbuke
- **Brojni izraz**
  - — argument može biti: brojni podatak, brojna promenljiva i numerička funkcija
  - — aritmetička operacija može biti: sabiranje (+), oduzimanje (—), množenje (*) i deljenje(/)
- **Aritmetička naredba**
  (brojna promenljiva) = (brojni izraz)
- **Azbučni podatak** je niska osnovnih simbola zapisana između znakova navoda.
- **Azbučna promenljiva** je X$ i Y$
- **Azbučni izraz:**
  - — argument može biti: azbučni podatak, azbučna

promenljiva i azbučna funkcija,
— operacija: dopisivanja (+).

- Naredba obrade azbučnih podataka
  ⟨azbučna promenljiva⟩=⟨azbučni izraz⟩

Ograničenje: ako se azbučna promenljiva sa leve strane znaka jednakosti javlja u azbučnom izrazu na desnoj strani znaka jednakosti, tada se ova promenljiva mora javiti kao prva sleva u azbučnom izrazu.

- Naredba izlaza

$$\text{PRINT}\left[\left\{\begin{matrix}\langle\text{izraz}\rangle\\ \text{AT}\langle\text{brojni izraz}\rangle\end{matrix}\right\}\left\{\begin{matrix}:\\ ;\end{matrix}\right\}\right]^{*}\left[\left\{\begin{matrix}\langle\text{izraz}\rangle\\ \text{AT}\langle\text{brojni izraz}\rangle\end{matrix}\right\}\left[\begin{matrix}:\\ ;\end{matrix}\right]\right]$$

- Naredba za algoritamski kraj programa: STOP
- Naredba za komentar u programu:
  ![⟨osnovni simbol⟩]*
- Naredba ulaza
  INPUT ⟨promenljiva⟩
- Numeričke funkcije:
  INT(b) — celobrojna funkcija, b-brojni izraz,
  RND — generisanje pseudo-slučajnih brojeva,
  PTR p — adresa promenljive p,
  VAL(a) — konverziona funkcija, a-brojni izraz koji definiše adresu polja u kojem se nalazi brojni izraz čija vrednost se uzima kao vrednost funkcije.
- Azbučna funkcija:
  CHR$(b) — konverzija dekadne vrednosti u ASCII-znak; b-brojni izraz.

Pitanja i zadaci za vežbu

1. Koristeći računar kao kalkulator izračunajte vrednosti aritmetičkih izraza:

   a) $2{,}5^2 - 3{,}1^2$ b) $\frac{4{,}2}{1{,}7} + \frac{3{,}5}{(-1{,}2)}$ c) $\frac{44{,}7}{35{,}1} : \frac{13{,}2}{4{,}7}$

2. Kako ćete izračunati vrednost aritmetičkog izraza

$$\frac{7{,}10\times10^{20}\cdot 6{,}25\times10^{25}}{1{,}37\cdot10^{22}}$$

   Objasnite koji zapis izraza dovodi do prekoračenja, a koji daje korektan rezultat. Zašto?
3. Sastaviti program koji za zadatu vrednost ugla u stepenima određuje i izdaje na ekranu vrednost ugla u radijanima.
4. Sastaviti program koji za dva uneta broja izdaje njihovu srednju vrednost.
5. Sastaviti program koji uneto ime grada izdaje od 10. pozicije 5-tog reda na ekranu.
6. Sastaviti program koji izdaje vašu adresu stanovanja u obliku koji je uobičajen u poštanskom saobraćaju.

6
RAZGRANATI PROGRAMI

Kod linijskih programa naredbe se izvršavaju jedna za drugom u zapisanom redosledu. Međutim, u mnogim zadacima postoji potreba da se promeni redosled izvršavanja naredbi u zavisnosti od izvesnih uslova. Ovo se postiže pomoću posebnih naredbi upravljanja, koje zovemo *naredbe prelaska*. Programi u kojima postoje naredbe prelaska zovu se *razgranati programi*. Karakteristika ovih programa jeste da se svaka naredba programa može izvršiti najviše jedanput. To znači da će se neke naredbe izvršiti jedanput, a neke naredbe se neće izvršiti u toku jednog izvravanja programa.

Naredba prelaska može biti· naredba uslovnog prelaska i bezuslovnog prelaska. U naredbi uslovnog prelaska javlja se relacijski izraz. Zato ćemo se prvo upoznati sa sintaksom i semantikom relacijskog izraza, a zatim ćemo se upoznati sa odgovarajućom naredbom prelaska.

6.1. Relacijski izraz

Relacijski izraz je elementarna konstrukcija u jeziku u kojoj su dve veličine postavljene u određenu relaciju. Ako veličine zadovoljavaju navedenu relaciju tada je vrednost izraza T (tačna) u protivnom vrednost izraza je ⊥ (netačna). Relacija je definisana relacijskim znakom. U sledećem tekstu upoznaćemo razne oblike relacijskih izraza.

Brojni relacijski izraz

Ako su izrazi u relacijskom izrazu brojni, onda je to *brojni relacijski izraz*. Brojni relacijski izraz se piše u obliku:

⟨brojni izraz⟩ ⟨relacijski znak⟩ ⟨brojni izraz⟩

gde relacijski znak može biti· jednako (=), manje (<) ili veće (>). Kada se promenljivim koje se javljaju u relacijskom izrazu dodele vrednosti, tada se relacijski izraz svodi na iskaz koji može biti tačan (u oznaci T) ili netačan (u oznaci ⊥).

Primer 1. Relacijski izraz

X=3

ima vrednost T ako je X jednako 3, a vrednost ⊥ ako nije jednako 3.

Primer 2. Relacijski izraz

A<B+1

ima vrednost T ako je A manje od B+1, a vrednost ⊥ ako je A jednako ili veće od B+1.

Primer 3. Relacijski izraz

INT(X+2)>10

ima vrednost T ako je najveći ceo broj jednak ili manji od X+2 veći od 10, a vrednost ⊥ ako ovaj uslov nije zadovoljen.

Azbučni relacijski izraz

Ako su izrazi u relacijskom izrazu azbučni, onda je to *azbučni relacijski izraz*. Relacijski azbučni izraz se piše u obliku:

EQ(azbučna promenjiva), (azbučna promenljiva)

Kao što se vidi u relacijskom izrazu se mogu koristiti samo azbučne promenljive. Prema tome, vrši se poređenje tekućih vrednosti azbučnih promenljivih. Kako su tekuće vrednosti azbučni podaci, a relacijski znak EQ znači relaciju jednakosti, to će relacijski izraz imati vrednost T ako su azbučni podaci jednaki, a ⊥ ako nisu jednaki. Dva azbučna podatka su jednaka ako su iste dužine i sleva nadesno sadrže iste znake.

Primer 4. Neka je X$="BEOGRAD", a Y$="ZAGREB", tada azbučni relacijski izraz

EQ X$, Y$

ima vrednost ⊥.

Tastaturni relacijski izraz

U nekim programima, naročito onim koji omogućuju razne igre sa računarom, pogodno je raspoznati situaciju kada je neki taster pritisnut od strane korisnika. Ovo se raspoznaje pomoću tastaturnog relacijskog izraza

KEY((brojni izraz))

koji ima sledeće značenje: odredi se celobrojna vrednost brojnog izraza i neka je to n, tada važi sledeće

$$KEY(n) = \begin{cases} T & \text{ako je pritisnut taster sa brojem n,} \\ \bot & \text{ako nije pritisnut taster sa brojem n.} \end{cases}$$

Svakom tasteru na tastaturi je pridružen po jedan ceo broj, prema tabeli 6.1.1. Tako, ako se u programu nalazi relacijski izraz KEY(1) tada će ovaj izraz imati vrednost T ako je u trenutku njegovog izračunavanja pritisnut taster za slovo A. Ako ovaj taster nije pritisnut u trenutku izračunavanja izraza KEY(1), tada će ovaj imati vrednost ⊥. Dakle, ovde je u pitanju relacija jednakosti broja n zapisanog u KEY(n) i broja pridruženog pritisnutom tasteru. Zato smo ovo i nazvali tastaturna relacija.

taster	n	taster	n	taster	n	taster	n	taster	n
A	1	L	12	W	23	2	34	=	45
B	2	M	13	X	24	3	35	.	46
C	3	N	14	Y	25	4	36	/	47
D	4	O	15	Z	26	5	37	ENTER	48
E	5	P	16	↑	27	6	38	BRK	49
F	6	O	17	↓	28	7	39	REPT	50
G	7	R	18	←	29	8	40	DEL	51
H	8	S	19	→	30	9	41	LIST	52
I	9	T	20	␣	31	;	42	SHIFT	53
J	10	U	21	Ø	32	:	43		
K	11	V	22	1	33	,	44		

Tabela 6.1.1. Brojevi pridruženi tasterima

6.2. Naredba uslovnog prelaska

Naredba koja omogućuje grananje u programu po vrednosti relacijskog izraza zove se *naredba uslovnog prelaska.* Ova naredba se piše u obliku

IF (relacijski izraz)(lista naredbi)

gde je lista naredbi niz naredbi međusobno razdvojenih sa dve tačke (:). Značenje ove naredbe je sledeće: Ako je relacijski izraz tačan, tada se izvršavaju naredbe, sleva na desno, navedene u listi naredbi, a zatim se prelazi na naredbu u sledećem programskom redu. Ako relacijski izraz nije tačan, tada se ne izvršavaju naredbe u listi naredbi već se prelazi na programski red koji sledi iza naredbe IF. Tako, naredba

IF A<B PRINT A,B

izdaje vrednosti A i B ako je A<B

Primer 1.

Sastaviti program koji za uneta dva broja X i Y izdaje tekst:

X JE MANJE OD Y, ako je X<Y
X JE JEDNAKO Y, ako je X=Y
X JE VEĆE do Y, ako je X>Y

Program se može zapisati u obliku:

```
10 INPUT X
20 INPUT Y
30 IF X<Y PRINT"X JE MANJE OD Y"
40 IF X=Y PRINT"X JE JEDNAKO Y"
50 IF X>Y PRINT"X JE VEĆE OD Y"
60 STOP
```

Izvršavanjem ovog programa može se dobiti sledeći protokol:

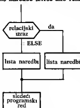
```
> RUN
?1
?5
X JE MANJE OD Y
> RUN
?4
?4
X JE JEDNAKO Y
> RUN
?7.2
?3.4
X JE VEĆE OD Y
> _
```

Praćenjem ovog protokola lako se zaključuje da se program korektno izvršava.

Izložena naredba IF daje mogućnost da se naredbe navedene u listi naredbi izvrše samo ako relacijski izraz ima vrednost T, odnosno navedene veličine su u zadatoj relaciji (sl. 6.2.1). Međutim, u programiranju je često pogodno da se izvesne naredbe izvrše ako relacijski izraz ima vred-



Sl. 6.2.1. Naredba IF



Sl. 6.2.2. Naredba IF . . . : ELSE



Sl. 6.2.3. Naredba IF . . . ELSE

nost T, a neke druge naredbe ako relacijski izraz ima vrednost ⊥, tj. navedene veličine nisu u relaciji (sl. 6.2.2). U ovom slučaju programski red može imati sledeću strukturu:

IF⟨relacijski izraz⟩⟨lista naredbi⟩:ELSE⟨lista naredbi⟩

Prvi deo programskog reda je isti kao u slučaju IF-naredbe, a drugi deo sadrži službenu reč ELSE iza koje se može navesti niz naredbi međusobno razdvojenih sa dve tačke. Tako, naredba

```
IF X<Y PRINT "X<Y":ELSE PRINT "NIJE X<Y"
```

Izdaje tekst X<Y, ako je X manje od Y, a tekst NIJE X<Y ako nije X manje od Y.

Nekada je pogodno izvršiti negaciju vrednosti relacijskog, tj. da se izvrši lista naredbi kada relacija nije zadovoljena (sl. 6.2.3). U ovom slučaju može se naredba IF pisati u obliku:

IF(relacijski izraz) [:]ELSE (lista naredbi)

Sada se naredbe navedene u listi naredbi izvršavaju ako relacijski izraz ima vrednosti ⊥. Na primer, naredba

IF X<Y ELSE PRINT"NIJE X<Y"

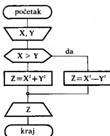
izdaje tekst NIJE X<Y ako je X jednako ili veće od Y. Kao što se vidi iz definicije naredbe, u ovom slučaju, dve tačke se mogu navesti ispred reči ELSE ali ne moraju.

Primer 2.

Sastaviti program koji za zadato X i Y izračunava Z na sledeći način:

$$Z = \begin{cases} X^2+Y^2 \text{ ako je } X \leq Y \\ X^2-Y^2 \text{ ako je } X > Y \end{cases}$$

Ovaj zadatak se može rešiti na više načina. Međutim, ovde je posebno pogodno iskoristiti naredbu IF ... ELSE, jer kada je uslov ispunjen (X>Y) tada treba izvršiti jednu naredbu




```
10 !PROGRAM G621
20 INPUT X
30 INPUT Y
40 IF X>Y Z=X*X—Y*Y.ELSE Z=X*X+Y*Y
50 PRINT Z
60 STOP
```

(Z=X*X—Y*Y), a kada uslov nije ispunjen drugu (Z==X*X+Y*Y). U svakom slučaju, bez obzira po kojoj naredbi je izvršeno izračunavanje treba izdati rezultat (Z). Izvršavanjem programa G621 može se dobiti sledeći protokol na ekranu:

```
> RUN
?6
?4
20
> RUN
?4
?6
52
> _
```

Pri prvom izvršavanju ulazni podaci su X=6, Y=4, pa kako je 6>4 izračunava se Y=6×6—4×4=20. U drugom izvršavanju uneti su podaci X=4, Y=6, pa u ovom slučaju nije 4>6 i izračunava se Y=4×4+6×6=52.

6.3. Naredba bezuslovnog prelaska

Mi smo upoznali naredbu uslovnog prelaska. Međutim, u programiranju je vrlo često korisno izvršiti bezuslovni prelazak. Za ovo postoji posebna naredba u programskim jezicima, koja se u slučaju računara GALAKSIJA piše u obliku

$$\text{GOTO}\begin{Bmatrix}\langle\text{brojni podatak}\rangle\\ \langle\text{brojna promenljiva}\rangle\end{Bmatrix}$$

Značenje ove naredbe je sledeće: Celobrojna vrednost brojnog podatka ili brojne promenljive uzima se kao broj reda u programu i vrši prelazak na ovako određen broj reda. Program se dalje izvršava od broja reda na koji je izvršen prelazak. Ako ovako određen broj reda ne postoji u programu javlja se greška i prekida dalje izvršavanje programa.

Primer.

Sastaviti program koji za uneto A, B i C izračunava vrednost Y po formuli:

$$Y=\frac{A^2}{B—C}$$

U ovom primeru postoji mogućnost da je B=C, odnosno B—C=0, dakle da dođe do deljenja nulom. U ovom slučaju, ako je A≠0 rezultat će biti nedefinisan, pa ćemo o ovome voditi računa u programu. Program ćemo zapisati u obliku:

```
10 !PROGRAM G631
20 INPUT A
30 INPUT B
40 INPUT C
50 IF B=C GOTO 90
60 Y=A*A/(B—C)
70 PRINT "Y=";Y
80 GOTO 100
90 PRINT "DELILAC JEDNAK NULI"
100 STOP
```

Ovaj program smo zapisali u ovom obliku da bismo ilustrovali primenu naredbe GOTO. U redu 50 smo primenili bezuslovni prelazak (GOTO 90) ako je B=C, a u redu 80 bezuslovni prelazak na red 100 u kojem se završava program. Izvršavanjem programa G631 može se dobiti sledeći protokol:

```
> RUN
?8
?6
?4
Y= 32
> RUN
?90
?54
?54
DELILAC JEDNAK NULI
> _
```

Pri prvom izvršavanju programa biće A=8, B=6 i C=4, tako da je B—C≠0 i program izračunava rezultat Y=32. U drugom izvršavanju biće A=90, B=54 i C=54, pa je B—C=0 i računar izdaje poruku da je delilac jednak nuli, a izračunavanje se ne vrši.

6.4. Programski stil

Već smo videli da za rešavanje jednog zadatka možemo zapisati program na različite načine. Svi ovi programi daju iste rezultate. Za ovakve programe kažemo da su *ekvivalentni programi*. Međutim, kao što u prirodnom jeziku istu misao možemo iskazati različitim rečenicama, od kojih su neke bolje, a neke lošije u pogledu jezičkog stila, tako se isto može govoriti i o boljim i lošijim programima u pogledu stila programiranja. Način na koji je program zapisan zvaćemo *programski stil* [3]. Naredbe uslovnog i bezuslovnog prelaska posebno utiču na stil programa. Zapravo, ove naredbe daju mogućnost korisniku da zapis programa učini vrlo nečitljivim, a to je onda loš programski stil. Program treba učiniti čitljivim, kao i bilo koji drugi tekst, tako da čitanje programa vršimo sleva nadesno i odozgo na dole. Preterana upotreba naredbi prelaska, naročito bezuslovnog prelaska, primorava korisnika da vrlo često traži broj reda od kojeg treba čitati program. Zato se treba truditi da zapisan

program bude u pogledu stila što bolji. Programi zapisani dobrim stilom lakše se testiraju, lakše se vrše izmene u programu do kojih u vreme eksploatacije programa može doći, a naročito su pogodniji za razmenu među korisnicima računara jer su čitljiviji. Kako postići dobar stil? Na ovo pitanje se može dati samo preporuka, koju će neki korisnici više, a neki manje uspešno sprovoditi u praksi. Program se sastoji od elementarnih struktura kakve su linijska, razgranata i ciklička struktura. Program komponovati tako da se lako svaka od ovih struktura uočava i da po pravilu postoji jedno mesto ulaska i jedno mesto izlaska iz ovakve strukture. Za programe koji se pišu po ovoj preporuci kaže se da su *dobro strukturirani programi* [4].

Nećemo se ovde dublje baviti problemom programskog stila. Dobar stil programiranja dosta zavisi i od mogućnosti programskog jezika. BASIC-jezik ne zadovoljava uslove za najbolji programski stil, kao što to zadovoljava, recimo, PASCAL [5]. Međutim, u okviru svakog programskog jezika mogu se pisati manje ili više dobro strukturirani programi. Mi ćemo kroz primere negovati što bolji programski stil, što naravno ne znači da će svaki primer biti zapisan na najbolji mogući način u pogledu stila.

Primer.

Program G631 u odeljku 6.3 je primer loše napisanog programa u pogledu stila. Naravno, u tom primeru je žrtvovan dobar stil da bi se ilustrovala primena naredbe GOTO. Program G631 bi bilo bolje zapisati u obliku:

```
10 !PROGRAM G641
20 INPUT A
30 INPUT B
40 INPUT C
50 IF B=C PRINT"DELILAC JEDNAK NULI":STOP
60 Y=A*A/(B—C)
70 PRINT"Y=";Y
80 STOP
```

Program je sada bez naredbe GOTO i lakše je čitljiv. U programskom redu 50 čitamo sve što računar treba da uradi u slučaju da je B=C. Ovo nije bio slučaj sa zapisom programa G631. Naravno, oba programa, G631 i G641, daju, za iste ulazne podatke, uvek isti rezultat. Dakle, to su ekvivalentni programi koji se međusobno razlikuju samo u stilu pisanja.

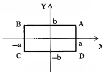
6.5. Rešeni zadaci

6.5.1. U pravouglom koordinatnom sistemu zadat je pravougaonik P sa temenima A, B, C i D (sl. 6.5.1). Sastaviti program koji za zadate koordinate tačke $M(x_0, y_0)$ određuje da li je tačka unutar ili na konturi, ili van pravougaonika P.

Rešenje:

Ako su zadate veličine a i b, tada je A(a, b), B(—a, b), C(—a, —b) i D(a, —b). Tačka $M(x_0, y_0)$ nalazi se unu-

tar ili na konturi pravougaonika ako je $|x_o| \leq a$ i $|y_o| \leq b$. Pošto ovo ispitivanje zahteva relaciju ma-



Sl. 6.5.1. Pravougaonik P

```
10 !PROGRAM G651
20 PRINT "UNETI A,B,X,Y:"
30 INPUT A:INPUT B:INPUT X:INPUT Y
40 IF X<0 X=—X
50 IF Y<0 Y=—Y
60 IF X>A PRINT"TAČKA JE VAN P.":STOP
70 IF Y>B PRINT"TAČKA JE VAN P.":STOP
80 PRINT"TAČKA JE UNUTAR P."
90 STOP
```

nje ili jednako, to će biti pogodnije u programu ispitivati da li je tačka van pravougaonika, a za to je potrebno da je $|x_o| > a$ ili $|y_o| > b$. Ovo je ostvareno u programu G651. Pošto je potrebna apsolutna vrednost x_o i y_o, to je u naredbama 40 i 50 promenjen znak ovim veličinama ako su negativne. U naredbama 60 i 70 ispituje se da li je tačka M van pravougaonika P i ako jeste daje se odgovarajući izveštaj, a ako nije tada je tačka M unutar ili na konturi pravougaonika P, pa se opet dobija odgovarajući izveštaj. Sledeći protokol prikazuje izvršavanje programa G651:

```
> RUN
UNETI A,B,X,Y:
?5
?2
?—4
?1
TAČKA JE UNUTAR P.
> RUN
UNETI A,B,X,Y:
?20
?10
?30
?—5
TAČKA JE VAN P.
> _
```

U prvom izvršavanju programa ulazni podaci su $a=5$, $b=2$, $x_o=-4$ i $y_o=1$, pa je tačka unutar pravougaonika P. U dru-

gom izvršavanju za a=20, b=10, x_o=30 i y_o=—5 tačka je van pravougaonika P.

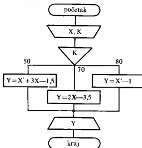
6.5.2. U ovom primeru želimo da demonstriramo mogućnost višestrukog grananja koristeći promenljivu u naredbi bezuslovnog prelaska (GOTO). U ovom cilju sastavićemo program koji za ulazne veličine X i K izračunava vrednost Y po formuli:

$$Y=\begin{cases} X^2+3X-1{,}5 & \text{ako je } K=70 \\ \text{ako je } K=50 & X^3-1 \\ 2X-3{,}5 & \text{ako je } K=80 \end{cases}$$

gde je K broj reda u programu u kojem se vrši odgovarajuće izračunavanje.

Rešenje:

Algoritamska šema za rešavanje ovog zadatka prikazana je na sl. 6.5.2. Kao što se vidi grananje se vrši u zavisnosti od unetog broja reda (K). Po ovoj algoritamskoj šemi je zapisan program G652.



Sl. 6.5.2. Algoritam za program G652

```
10 !PROGRAM G652
20 PRINT"UNESITE X", :INPUT X
30 PRINT"UNESITE BROJ REDA", :INPUT K
40 GOTO K
50 Y=X*X+3*X—1.5
60 PRINT"Y=";Y:STOP
70 Y=2*X—3.5:GOTO 60
80 Y=X*X*X—1:GOTO 60
```

Izvršavanjem programa G652 za ulaznu vrednost X=2 i sve tri vrednosti K dobija se sledeći protokol:

```
> RUN
UNESITE X?2
UNESITE BROJ REDA?50
Y=8.5
> RUN
UNESITE X?2
UNESITE BROJ REDA?70
Y= .5
> RUN
UNESITE X?2
UNESITE BROJ REDA?80
Y= 7
> _
```

6.5.3. Da bismo ilustrovali tastaturni relacijski izraz rešićemo zadatak 6.5.2. u nešto drugačijoj postavci. Sastaviti program koji izračunava vrednost funkcije Y(X) u zavisnosti od toga koji je taster pritisnut na tastaturi, tako da je

$$Y=\begin{cases} X^2+3X-1,5 & \text{ako je pritisnut taster 1} \\ 2X-3,5 & \text{ako je pritisnut taster 2} \\ X^3-1 & \text{ako je pritisnut taster 3} \end{cases}$$

Rešenje: Program se može zapisati u obliku

```
10 !PROGRAM G653
20 PRINT"UNESITE X"; :INPUT X
30 IF KEY(33)Y=X*X+3*X—1.5:GOTO 70
40 IF KEY(34)Y=2*X—3.5:GOTO 70
50 IF KEY(35)Y=X*X*X—1 GOTO 70
60 GOTO 30
70 PRINT"Y=";Y:STOP
```

U programu G653 izvrši se unošenje veličine X, a zatim, u naredbama 30, 40 i 50 vrši se izračunavanje veličine Y u zavisnosti od toga koji je taster na tastaturi pritisnut. Naredba u redu 60 (GOTO 30) obezbeđuje cikličko ponavljanje ispitivanja da li je jedan od navedenih tastera pritisnut. U ovako obrazovanom ciklusu se vrši čekanje da korisnik pritisne jedan od tastera (1, 2 ili 3). Kada korisnik pritisne jedan od ovih tastera izvrši se odgovarajuće izračunavanje i prede na naredbu izlaza (70). Izvršavanjem programa G653 može se dobiti sledeći protokol:

```
> RUN
UNESITE X?2
Y=8.5
> RUN
UNESITE X?2
Y= .5
> RUN
UNESITE X?2
Y= 7
```

Pri izvršavanju smo uzeli istu vrednost za X kao i u zadatku 6.5.2, a zatim koristili redom tastere 1, 2 i 3. Naravno i izračunata vrednost Y je ista. Protokol je kraći jer se ne vrši unošenje broja reda.

Kratak izvod

- Relacijski izraz može biti brojni, azbučni ili tastaturni.
- Naredba uslovnog prelaska može imati sledeće oblike:

IF(relacijski izraz)(lista naredbi)[:ELSE(lista naredbi)]
IF(relacijski izraz) [:]ELSE (lista naredbi)

- Naredba bezuslovnog prelaska

$$\text{GOTO}\begin{Bmatrix}\langle\text{brojni podatak}\rangle\\ \langle\text{brojna promenljiva}\rangle\end{Bmatrix}$$

- Dobar programski stil se ogleda u pisanju dobro struktuiranih programa.

Pitanja i zadaci za vežbu

1. Kritički ocenite dobre i loše osobine korišćenja naredbe bezuslovnog prelaska, a naročito kada se koristi u obliku GOTO p, gde je p brojna promenljiva.
2. Za zadata tri međusobno različita broja x, y i z na ulazu, sastaviti program koji izdaje najmanji od zadatih brojeva.
3. Zadata su tri broja a, b i c. Sastaviti program koji utvrđuje da li zadati brojevi mogu biti dužine stranica trougla.
4. Sastaviti program koji određuje rešenja sistema jednačina

$$a_1x+b_1y=c_1$$
$$a_2x+b_2y=c_2$$

5. Sastaviti program koji za zadati poluprečnik izračunava obim ili površinu kruga prema želji korisnika programa
6. Sastaviti program koji za zadati ceo broj na ulazu utvrđuje da li je to pozitivan ili negativan broj i da li je paran ili neparan.
7. Sastaviti program koji za zadato a, b i c određuje karakter rešenja kvadratne jednačine

$$ax^2+bx+c=0.$$

7

PROGRAMI SA CIKLUSIMA

7.1. Organizacija programskog ciklusa

Ako se niz naredbi u programu može izvršiti više puta u toku jednog izvršavanja programa, tada se kaže da ove naredbe obrazuju *programski ciklus*. Među naredbama koje čine ciklus mora postojati barem jedna naredba koja omogućuje izlazak iz programskog ciklusa. To je obično naredba uslovnog prelaska, tako da se za jednu vrednost uslova nastavlja izvršavanje ciklusa, a za drugu vrednost vrši se izlazak iz ciklusa. Uslov pod kojim se izlazi iz ciklusa zove se *izlazni kriterijum* ciklusa. Prema vrsti izlaznog kriterijuma programski ciklus može biti:

— *brojački*, ako je izlazni kriterijum broj ponavljanja ciklusa,

— *iterativan*, ako je izlazni kriterijum dostignuta tačnost u procesu računanja,

— *beskonačan*, ako se ciklus može ponavljati neograničen broj puta.

U sledećem izlaganju mi ćemo se upoznati sa navedene tri vrste programskih ciklusa. Za organizaciju ovih ciklusa nisu nam potrebne nove naredbe BASIC-jezika.

Beskonačni programski ciklusi

Već smo rekli da se beskonačni programski ciklus može ponavljati neograničen broj puta. Kod ovakvih ciklusa ne mogu nastati uslovi u samom procesu računanja koji će dovesti do izlaska iz ciklusa. Naravno, nema smisla dozvoliti neograničeno izvršavanje programskog ciklusa, jer se ovakvi ciklusi ne bi završili sve dok bi računar bio u radnom stanju. Ovakvi ciklusi se ipak mogu javiti u programu ili po želji programera ili pak usled greške u izlaznom kriterijumu ciklusa. Izvršavanje ovakvih ciklusa korisnik može prekinuti posebnom tastaturnom komandom (taster BRK).

Primer 1.

Primer 2 u odeljku 5.6 možemo rešiti tako da se izračunavaju vrednosti polinoma

$$P(x)=x^3-2{,}5\times x^2+4{,}2$$

za proizvoljan broj vrednosti argumenta x. Program se može zapisati u obliku:

```
10 !VREDNOST POLINOMA
20 PRINT"VREDNOST ARGUMENTA";
30 INPUT X
40 P=X*X*X—2.5*X*X+4.2
50 PRINT"VREDNOST POLINOMA JE";P
60 GOTO 20
```

Kao što se vidi program, u poslednjem programskom redu, sadrži naredbu bezuslovnog prelaska na početak programa. Pošto unutar programa ne postoje naredbe kojima se može završiti program ili izaći van naredbi koje se ponavlja-ju, to je na ovaj način obrazovan beskonačan programski ciklus. Izvršavanjem programa može se dobiti sledeći protokol:

```
> RUN
VREDNOST ARGUMENTA?1.5
VREDNOST POLINOMA JE 1.95
VREDNOST ARGUMENTA?4
VREDNOST POLINOMA JE 28.2
VREDNOST ARGUMENTA?
BREAK 30
> _
```

Dakle, program je izvršen za $x=1{,}5$ i $x=4$, a zatim je izdat zahtev za unošenje sledeće vrednosti za argument. Naravno, mogla bi da se unese nova vrednost argumenta i tako nastavi neograničen broj puta. U našem primeru pritisnut je taster BRK i prekinuto izvršavanje programa u naredbi sa brojem 30, što pokazuje izveštaj BREAK 30.

Brojački programski ciklus

Kod ove vrste programskih ciklusa kriterijum za izlazak iz ciklusa je broj ponavljanja ciklusa. Pri programiranju ovakvih ciklusa vrlo česta greška, naročito kod početnika, je da se programski ciklus ponavlja jedanput manje ili više nego što je to potrebno. Zato programske cikluse treba pažljivo testirati. Kod testiranja treba proveriti sledeće slučajeve:

— programski ciklus se ne izvršava, ako takav slučaj treba da je obuhvaćen zadatkom,
— programski ciklus se izvršava jedanputa i
— programski ciklus se izvršava dva puta.

Posle ove provere ciklus će se izvršavati tačno i za proizvoljno zadat broj ponavljanja.

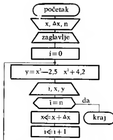
Primer 2.

Prethodni zadatak izračunavanja vrednosti polinoma, rešimo tako da se izdaju vrednosti polinoma

$$P(x)=x^3-2{,}5\cdot x^2+4{,}2$$

za vrednosti nezavisno promenljive $x_i=x_o+i\cdot\Delta x$, $i=0, 1, \ldots, n$.

U ovom zadatku ulazne veličine su x_o, Δx i n. Algoritamska šema data je na sl. 7.1.1. Kao što se vidi u ciklusu se izračunava vrednost polinoma, a zatim izračunata vrednost izdaje. Izlazni kriterijum iz ciklusa je ispitivanje da li je i=n. Ako je izlazni kriterijum zadovoljen, tada se program završava, a ako nije vrši se uvećanje vrednosti argumenta za priraštaj Δx i prebrojava se broj izračunatih vrednosti polinoma (promenljiva i). Prema ovom algoritmu je zapi-



Sl. 7.1.1. Izačunavanje vrednosti polinoma

```
10 !PROGRAM G711
15 !VREDNOST POLINOMA
20 PRINT"UNESITE X0, DELTA X I N:"
30 INPUT X:INPUT D:INPUT N
40 PRINT" I          X          Y(X)":PRINT
50 I=0
60 Y=X*X*X—2.5*X*X+4.2
70 PRINT I, X, Y
80 IF I=N STOP
90 X=X+D:I=I+1:GOTO 60
```

san i program. Izvršavanjem programa može se dobiti sledeći protokol:

```
> RUN

UNESITE X0, DELTA X I N:
?1.5
?0.5
?7
I          X          Y(X)

0          1.5        1.95
1          2          2.2
2          2.5        4.2
3          3          8.7
4          3.5        16.45
5          4          28.2
6          4.5        44.7
7          5          66.7
> _
```

Kod ove vrste programskih ciklusa kriterijum za izlazak iz ciklusa je dostignuta tačnost u toku računanja. Ovakvi programski ciklusi najčešće se koriste kod programiranja iterativnih numeričkih postupaka. Jedan prolazak kroz programski ciklus odgovara jednoj iteraciji u numeričkom postupku. Izlazni kriterijum može biti uslov da su razlike između vrednosti izračunatih u tekućem iterativnom koraku i vrednosti iz prethodnog koraka dovoljno male.

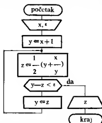
Primer.

Sastaviti program koji za zadato x izračunava $\sqrt{x}$, po Njutnovoj iterativnoj formuli

$$x_{i+1}=\frac{1}{2}\left(x_i+\frac{x}{x_i}\right),\quad i=0,\ 1,\ 2,\ \ldots$$

gde je $x_0=x+1$. Proces računanja prekinuti kada se dostigne zadata tačnost ε, tako da je $x_i - x_{i+1} < \varepsilon$.

Ovde su ulazne veličine x i ε, a izlazna veličina izračunat kvadratni koren iz x. Po ovoj metodi to će biti vrednost x_{i+1} izračunata u poslednjoj iteraciji. Algoritamska šema za rešavanje ovog zadatka data je na sl 7.1.2. Treba uočiti da se za navedeno iterativno računanje mogu koristiti dve promenljive, prva od njih (y) predstavlja prethodnu, a druga



Sl. 7.1.2. Izračunavanje kvadratnog korena

```
10 PROGRAM G711
20 PRINT"KVADRATNI KOREN IZ ";:INPUT X
30 PRINT"SA TAČNOSĆU ";:INPUT E
40 Y=X+1
50 Z=(Y+X/Y)/2
60 IF Y—Z<E PRINT"IZNOSI";Z:STOP
70 Y=Z:GOTO 50
```

(z) sledeću vrednost u iterativnom računanju. Na ovaj način, broj iterativnih koraka ne utiče na broj promenljivih koje se javljaju za vreme računanja, kao što je to slučaj u uobičajeno matematičkoj notaciji, gde se javljaju promenljive x_0, x_1, x_2 ... Uočimo da je kod iterativnih ciklusa broj ponavljanja ciklusa nepoznat pre izvršavanja programa. Ovaj broj zavisi od brzine konvergencije interativnog postupka i zadate tačnosti (ε). Program G711 napisan je prema algoritamskoj šemi na sl. 7.1.2. U programu je dodat tekst koji objašnjava ulazne i izlazne veličine. Ovaj tekst nećemo navoditi u algoritamskim šemama, jer nepotrebno opterećuje šemu, a nije bitan za razumevanje postupka. Izvršavanjem programa G711 dobija se sledeći protokol:

```
> RUN
KVADRATNI KOREN IZ ?2
SA TAČNOŠĆU ?0.0001
IZNOSI 1.41421
> _
```

Važni zaključci iz ovog primera

- broj ponavljanja iterativnog ciklusa ne može se unapred predvideti,
- u programima treba težiti da se javi što manji broj promenljivih, jer to smanjuje angažovanje memorijskog prostora.

7.2. Naredbe programskog ciklusa

Kako se u programiranju vrlo često koriste brojački programski ciklusi, to u BASIC-jeziku postoje posebne naredbe koje omogućuju lako programiranje ovakvih ciklusa. Struktura ovakvog ciklusa je sledeća

```
FOR i=v TO k [STEP p]
   ┌──────────────┐
   │ telo ciklusa │
   └──────────────┘
NEXT i
```

Za naredbe koje se nalaze unutar ciklusa kažemo da čine *telo ciklusa*. Telo ciklusa se nalazi između naredbe FOR i NEXT. U naredbi FOR, kojom počinje ciklus, nalaze se sledeće veličine:

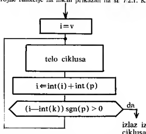
i — kontrolna promenljiva koja može biti bilo koja brojna promenljiva,

v — početna vrednost kontrolne promenljive i može se pisati u obliku brojnog izraza

k — krajnja vrednost kontrolne promenljive i može biti brojni izraz,

p — priraštaj kontrolne promenljive i može biti brojni izraz.

Ove veličine određuju izvršavanje ciklusa i to na sledeći način: Naredbe koje čine telo ciklusa izvršavaju se prvi put za vrednost kontrolne promenljive i=v, drugi put za i=v+p, treći put za i=v+2 · p itd. sve dok vrednost kontrolne promenljive ne dostigne krajnju vrednost k. Ako se priraštaj p ne navede podrazumeva se da je p=1. Sve ovo važi ako su v, k i p celobrojne veličine, a tako ih treba u programu i koristiti. Međutim, ako korisnik zapiše ove veličine u obliku brojnih izraza čije vrednosti nisu celobrojne, tada će sistem određivati celobrojne vrednosti pomoću celobrojne funkcije na način prikazan na sl. 7.2.1. Kao što



Sl. 7.2.1. Funkcija naredbe za programski ciklus

se vidi prvi prolaz kroz ciklus biće za navedenu početnu vrednost v, a zatim će kontrolna vrednost (i) dobijati samo celobrojne vrednosti prema sl. 7.2.1. Ako želimo da odredimo broj ponavljanja ciklusa, tada to možemo učiniti na sledeći način: Odredimo broj m

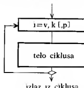
$$m = \mathrm{int}\left(\frac{\mathrm{int}(k) - \mathrm{int}(v)}{\mathrm{int}(p)}\right)$$

onda je broj ponavljanja ciklusa:

$$n = \begin{cases} m+1, & \text{ako je } m \geqslant 0 \\ \infty, & \text{ako je } m < 0 \end{cases}$$

Kao što se vidi ciklus se može izvršiti jedanputa, dvaputa, ... itd. konačan broj puta ako je $m \geqslant 0$. Međutim, ako je $m < 0$ tada se ciklus izvršava neograničen broj puta, dakle, postaje bezkonačan ciklus. Takođe je važno uočiti da se ciklus mora izvršiti bar jedanput.

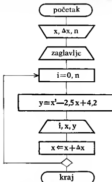
Da bismo uprostili grafički prikaz brojačkih programskih ciklusa, koje programiramo pomoću naredbi FOR i NEXT, koristićemo grafički zapis dat na sl. 7.2.2. Priraštaj ciklusa p ne mora se pisati ako je jednak jedinici.



Sl. 7.2.2. Grafički prikaz brojačkog ciklusa

Primer 1.

Ranije rešen primer izračunavanja vrednosti polinoma primenom naredbe uslovnog prelaska, sada možemo rešiti korišćenjem naredbe za programski ciklus. U sluča-



Sl 7.2.3. Algoritam za program G721

```
10 !PROGRAM G721
15 !VREDNOST POLINOMA
20 PRINT"UNESITE X0, DELTA X I N:"
30 INPUT X:INPUT D:INPUT N
40 PRINT" I        X          Y(X)":PRINT
50 FOR I=0 TO N
60 Y=X*X*X-2.5*X*X+4.2
70 PRINT I, X, Y
80 X=X+D
90 NEXT I
100 STOP
```

ju korišćenja naredbe uslovnog prelaska program je sastavljen prema algoritmu na sl. 7.1.1. U ovom slučaju algoritam je prikazan na sl. 7.2.3, a odgovarajući program je G721.

Prva naredba ciklusa nalazi se u redu 50, a poslednja u redu 90. Između ovih redova nalaze se naredbe koje čine telo ciklusa. Kako je priraštaj 1 to nije naveden, a ciklus će se izvršiti N+1 puta. Izvršavanjem programa G721 dobija se isti izveštaj kao i pri izvršavanju programa G711.

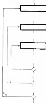
Međusobni odnos ciklusa

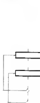
Složeniji programi sadrže često veći broj ciklusa. Ako su ciklusi pisani sa naredbama FOR i NEXT, tada se u složenim programima mogu javiti kao

- linijska kompozicija ciklusa i/ili
- koncentrična kompozicija ciklusa.

Kod linijske kompozicije ciklusi se ređaju jedan za drugim, dok se kod koncentrične kompozicije ciklusi nalaze jedan unutar drugog. Tako, ciklusi $c_1, c_2, \ldots, c_n$, na sl. 7.2.4. čine linijsku kompoziciju, a ciklusi $c_1, c_2, \ldots, c_n$, na sl. 7.2.5, čine koncentričnu kompoziciju ciklusa. Ovakve kompozicije se mogu kombinovati, ali se ne sme dozvoliti presecanje ciklusa, kako je to prikazano na sl. 7.2.6.


Sl. 7.2.4

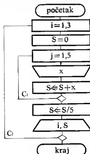

Sl. 7.2.5.


Sl. 7.2.6.

Primer 2

U ovom primeru videćemo upotrebu koncentrične kompozicije ciklusa. Neka su zadata tri niza brojeva i svaki niz sadrži 5 brojeva. Izračunati srednju vrednost svakog niza.

Na sl. 7.2.7. data je algoritamska šema, po kojoj je napisan program G722. Kao što se vidi na sl. 7.2.7. postoje dva ciklusa c_1 i c_2 koji čine koncentričnu kompoziciju, jer se ciklus c_1 nalazi unutar ciklusa c_2. Za ciklus c_2 se kaže da je spoljašnji, a ciklus c_1 unutrašnji. U unutrašnjem ciklusu se vrši unošenje elemenata niza i njihovo sumiranje. Sumiranje se vrši dodavanjem unetog elementa na prethodnu vrednost promenljive S, s tim što se promenljivoj S dodeli nula pre ulaska u unutarnji ciklus. Ovo je važan detalj ovog programa, jer da nije postavljeno S=0, pre ulaska u unutarnji ciklus, elementi novog niza bili bi sumirani na pret-



sl. 7.2.7.

```
10 !PROGRAM G722
20 FOR I=1 TO 3
30 S=0
40 PRINT"UNETI VREDNOSTI";I;" NIZA:"
50 FOR J=1 TO 5
60 INPUT X
70 S=S+X
80 NEXT J
90 S=S/5
100 PRINT"SREDNJA VREDNOST:";S
110 NEXT I
120 STOP
```

hodno određenu sumu, što bi dovelo do pogrešnog rezultata. Izvršavanjem programa G722 može se dobiti sledeći protokol:

```
> RUN
UNETI VREDNOSTI 1. NIZA.
?12.85
?14.22
?11.72
?12.01
?10.86
SREDNJA VREDNOST: 12.332
UNETI VREDNOSTI 2. NIZA·
? _
```

U protokolu je prikazan izveštaj o unošenju i izračunavanju srednje vrednosti 1. niza od 5 brojeva a zatim se izveštaj nastavlja za 2. i 3. niz na sličan način.

7.3. Nizovi podataka

U mnogim oblastima primene računara javlja se veliki broj ulaznih podataka, međurezultata i rezultata u programu. Često je pogodno, ovakvim skupovima podataka dodeliti zajedničko ime, a ne imenovati svaki podatak pojedinačno. Ovakav skup podataka, kojem se dodeljuje zajedničko ime u BASIC-jeziku, zove se *niz podataka*.

Svi podaci koji ulaze u niz moraju biti isti po vrsti, što znači da svi podaci mogu biti brojni podaci i niz se tada zove *brojni niz*, ili azbučni podaci, a niz se tada zove *azbučni niz*.

7.3.1. Niz brojnih podataka

Kod računara GALAKSIJA postoji samo jedan brojni niz i nosi ime A. To je *jednodimenzionalni niz*, čiji broj elemenata zavisi od raspoloživog memorijskog prostora za niz. Zapravo, slobodan memorijski prostor se koristi za smeštaj BASIC-programa, a deo memorije koji se ne zauzme programom može se koristiti za brojni niz. Zato se brojni niz ne dimenzioniše posebnom naredbom, već se raspoloživi memorijski prostor popunjava elementima niza sve dok postoji slobodan prostor u memoriji. U trenutku kada se ceo prostor ispuni, a zahteva se registrovanje novog elementa niza računar javlja grešku (SORRY) i prekida dalje izvršavanje programa. Jedan elemenat niza zahteva 4 bajta u memoriji, a broj elemenata će zavisiti od raspoloživog memorijskog prostora. Tako se korisniku može dogoditi, ako ima dugačak program, da može da registruje mali broj elemenata brojnog niza, a ako ima vrlo mali program može lako registrovati i preko hiljadu elemenata. Operacije nad elementima niza zadaju se na isti način kao i za brojni promenljivu.

Da bi se ukazalo na elemenat niza, iza imena niza (A) navodi se *indeks* koji ukazuje na elemenat niza. Indeks se piše između zagrada, dakle, u obliku

A(n)

gde n može biti i brojni izraz, čija celobrojna vrednost definiše elemenat niza. Celobrojna vrednost indeksa može biti 0, 1, 2,..., a ovi indeksi određuju *indeksne promenljive* A(o), A(1), A(2), ... čije vrednosti su redom elementi niza: nulti, prvi, drugi itd. Indeksna promenljiva se može javiti menljiva kojoj se dodeljuje brojna vrednost. Sledeći promenljiva kojoj se odeljuje brojna vrednost. Sledeći program ilustruje primenu niza:

```
10 I=0
20 A(I)=I
30 PRINT I, A(I)
40 I=I+1
50 GOTO 20
```

U ovom programu elementima niza se dodeljuju vrednosti indeksa, tako da je A(0)=0, A(1)=1, A(2)=2, itd. U naredbi 30 vrši se izdavanje vrednosti indeksa (I) i odgovarajuće vrednosti indeksne promenljive A(I). Program obrazuje beskonačan ciklus i angažovaće sav raspoloživi memorijski prostor za brojni niz. Prema tome, program će se izvršavati sve dok se raspoloživi memorijski prostor ne iskoristi za registrovanje elemenata niza. Kada se sav prostor iskoristi računar će javiti grešku (SORRY) i prekinuti izvršavanje programa.

Primer

Sastaviti program koji za N unetih brojeva određuje koliko brojeva je manje od poslednjeg unetog broja. Program se može zapisati u obliku:

```
10 PRINT"UNESITE BROJ ELEMENATA ";:INPUT N
20 PRINT"UNESITE ELEMENTE NIZA:"
30 FOR I=1 TO N
40 INPUT A(I)
50 NEXT I
60 M=0
70 FOR I=1 TO N—1
80 IF A(I)<A(N) M=M+1
90 NEXT I
100 PRINT"NIZ SADRŽI";M;" ELEMENTA MANJA OD",
    A(N)
110 STOP
```

Program sadrži dva ciklusa. U prvom ciklusu se vrši unošenje elemenata koji se registruju u obliku brojnog niza od A(1) do A(N). U drugom ciklusu se vrši prebrojavanje onih elemenata niza koji su manji od poslednjeg unetog elementa, a to je elemenat A(N). Izvršavanjem programa može se dobiti sledeći protokol:

```
> RUN
UNESITE BROJ ELEMENATA ?6
UNESITE ELEMENTE NIZ:
?2
?1
?5
?7
?3
?4
NIZ SADRŽI 3 ELEMENTA MANJA OD 4
> _
```

7.3.2. Niz azbučnih podataka

U BASIC-jeziku za računar GALAKSIJU može se definisati jedan azbučni niz sa imenom X$. To je jednodimenzionalni niz, čiji se broj elemenata definiše naredbom

ARR$(n)

gde je n brojni izraz čija celobrojna vrednost određuje broj elemenata azbučnog niza X$. Ova naredba se može nalaziti bilo gde u programu, ali pre naredbe u kojoj se koriste elementi niza. Dobra programerska praksa je da se ova naredba piše na početku programa.

Svaki elemenat azbučnog niza može biti azbučni podatak od najviše 16 znakova. Jedan znak zauzima jedan bajt u memoriji. Na elemenat niza se ukazuje indeksom, kao i kod brojnih promenljivih, tako da se azbučna indeksna promenljiva piše u obliku

X$(m)

gde je m brojni izraz, čija celobrojna vrednost definiše elemenat niza. Azbučne indeksne promenljive se mogu koristiti na isti način kao i obične azbučne promenljive. Tako je u programu

```
10 INPUT N
20 ARR$(N)
30 FOR I=0 TO N
40 INPUT X$(I)
50 NEXT I
60 FOR I=0 TO N
70 PRINT I, X$(I)
80 NEXT I
90 STOP
```

niz dimenzionisan u naredbi broj 20. Veličina niza je određena brojem N sa ulaza. U ciklusu od 30. do 50. naredbe unosi se N+1 elemenat azbučnog niza i dodeljuje indeksnim promenljivim X$(0), X$(1),... X$(N). U ciklusu koji sledi (od 60 do 80 naredbe) vrši se izdavanje elemenata unetog niza. Uočimo, da se naredba za dimenzionisanje (red 20 mora nalaziti posle naredbe u kojoj se definiše vrednost promenljive N (naredba 10), a pre naredbe u kojoj se koriste elementi niza (naredba 40)

Jedanput definisan azbučni niz naredbom ARR$ ostaje u memoriji i posle primene komandi NEW ili OLD, a definicija se može učiniti nevažećom samo naredbom ARR$(—1).

Primer.

Sastaviti program koji za N unetih reči određuje koliko se puta javlja poslednja uneta reč u nizu od N unetih reči. Program se može zapisati u obliku:

```
10 PRINT"UNESITE BROJ ELEMENATA ";:INPUT N
20 PRINT"UNESITE ELEMENTE NIZA:"
25 ARR$(N)
30 FOR I=1 TO N
40 INPUT X$(I)
50 NEXT I
60 M=0
70 FOR I=1 TO N
80 IF EQ X$(I), X$(N) M=M+1
90 NEXT I
100 PRINT"NIZ SADRŽI";M," REČI ",X$(N)
```

Program sadrži dva ciklusa. U prvom se vrši unošenje N reči koje se dodeljuju azbučnim indeksnim promenljivim X$(1), X$(2), ..., X$(N). U drugom ciklusu se vrši prebrojavanje reči koje su iste sa vrednošću indeksne promenljive X$(N). Izvršavanjem programa može se dobiti sledeći protokol:

```
> RUN
UNESITE BROJ ELEMENATA ?5
UNESITE ELEMENTE NIZA:
?BEOGRAD
?ČAČAK
?KRAGUJEVAC
?NIŠ
?ČAČAK
NIZ SADRŽI 2 REČI ČAČAK
> _
```

7.4. Rešeni zadaci

7.4.1 Sastaviti program koji za zadati ulog, godišnji interes u procentima i broj godina izračunava godišnju kamatu i sumu na kraju svake godine obračunavajući složeni interes.

Rešenje: U programu ćemo predvideti unošenje uloga (U), interesa (P) i broja godina (G). Zatim ćemo obrazovati ciklus u kojem će se izračunavati kamata i suma na kraju svake godine. Program se može zapisati u obliku:

```
10 !PROGRAM G741
20 PRINT"UNESITE ULOG ";:INPUT U
30 PRINT"UNESITE INTERES (U %)";:INPUT P
40 PRINT"UNESITE BROJ GODINA ";:INPUT G
45 PRINT"GODINA   ULOG   KAMATA   SUMA"
```

```
50 FOR I=1 TO G
60 K=(P*U)/100
70 S=U+K
80 PRINT I;".",U,K,S
90 U=S
100 NEXT I
110 STOP
```

U ciklusu je predviđeno i izdavanje rezultata: redni broj godine, ulog na početku godine, kamata i suma uštede na kraju svake godine. Izvršavanjem programa može se dobiti sledeći protokol:

```
> RUN
UNESITE ULOG ?100000
UNESITE INTERES (U%) ?28
UNESITE BROJ GODINA ?5
GODINA    ULOG      KAMATA     SUMA
1.        100000    28000      128000
2.        128000    35840      163840
3.        163480    45875.2    209715
4.        209715    58720.2    268435
5.        268435    75161.9    343597
```

Iz protokola se vidi da ulog od 100000 din. sa interesom od 28% posle 5 godina daje sumu od 343597 din.

7.4.2. Sastaviti program koji niz brojeva $a_1, a_2, \ldots, a_n$, $n \leq 15$, uređuje u nerastući niz brojeva.

Rešenje: Uređenje niza brojeva može se izvršiti upoređivanjem svaka dva susedna broja u nizu

$$a_i \geq a_{i+1}, \quad i=1, 2, \ldots, n-1$$

pri ovome mogu nastati dva slučaja:

a) Brojevi a_i i a_{i+1} zadovoljavaju zahtevanu relaciju, tako da je $a_i \geq a_{i+1}$. Takve članove niza ne treba premeštati.

b) Brojevi a_i i a_{i+1} ne zadovoljavaju zahtevanu relaciju pa takvim brojevima treba promeniti mesta u nizu. Razmena mesta članova niza može se izvršiti sledećim postavljanjem

$$p \Leftarrow a_i$$
$$a_i \Leftarrow a_{i+1}$$
$$a_{i+1} \Leftarrow p$$

c) Niz je uređen u nerastući niz brojeva, ako svaka dva susedna broja u nizu zadovoljavaju zahtevanu relaciju, tj. ako je $a_i \geq a_{i+1}$, $i=1,2, \ldots, n-1$. Ovo se kontroliše vrednošću promenljive k, tako što se pre prolaska kroz niz postavi $k=0$, a zatim ako dođe do promene mesta članovima niza, postavlja se $k=1$. Na ovaj način, ako je, po izlasku iz unutrašnjeg ciklusa, vrednost promenljive k jednaka jedinici, tada se uređenje niza nastavlja, a ako je jednaka nuli, niz je uređen. Program G742 je sastavljen prema izloženoj ideji za uređenje niza brojeva.

```
10 'PROGRAM G742
20 PRINT"BROJ ELEMENATA NIZA ";:INPUT N
30 PRINT"UNESITE ELEMENTE NIZA."
40 FOR I=1 TO N
50 INPUT A(I)
60 A(N+I)=A(I)
70 NEXT I
80 K=0
90 FOR I=1 TO N—1
100 IF A(I)<A(I+1)ELSE GOTO 150
110 P=A(I)
120 A(I)=A(I+1)
130 A(I+1)=P
140 K=I
150 NEXT I
160 IF K=I GOTO 80
170 PRINT"ELEMENT NEUREĐ UREĐEN"
180 PRINT"  NIZA     NIZ      NIZ"
190 FOR I=1 TO N
200 PRINT I, A(N+I),A(I)
210 NEXT I
220 STOP
```

Izvršavanjem programa G742 za 8 članova neuređenog niza dobija se sledeći izveštaj:

```
> RUN
BROJ ELEMENATA NIZA ?8
UNESITE ELEMENTE NIZA:
?67
?45
?23
?12
?90
?43
?21
?34

ELEMENT  NEUREĐ  UREĐEN
  NIZA     NIZ     NIZ

   1       67      90
   2       45      67
   3       23      45
   4       12      43
   5       90      34
   6       43      23
   7       21      21
   8       34      12
> _
```

7.4.3. Sastaviti program koji omogućuje unošenje imena gradova sa odgovarajućim poštanskim brojevima za sve glavne gradove republika i pokrajina u Jugoslaviji. Omogućiti da se ovaj program koristi tako, što će za uneti grad program izdavati poštanski broj grada.

Rešenje. Program se može zapisati u obliku:

```
10 PROGRAM G743
20 ARR$(7)
30 FOR I=0 TO 7
40 PRINT"NAZIV GRADA ";:INPUT X$(I)
50 PRINT"POŠTANSKI BROJ ";:INPUT A(I)
60 NEXT I PRINT
70 PRINT"POŠTANSKI BROJ GRADA ";:INPUT X$
80 FOR I=0 TO 7
90 IF EQ X$(I),X$ GOTO 130
100 NEXT I
110 PRINT"U MEMORIJI NE POSTOJI GRAD SA
    IMENOM ";X$
120 GOTO 70
130 PRINT"GRAD ",X$," IMA POŠTANSKI BROJ"; A(I)
140 GOTO 70
```

U programu je dimenzionisan azbučni niz od 8 elemenata (od 0 do 7). U prvom ciklusu (od 30. do 60. naredbe) se vrši unošenje imena gradova i odgovarajućih poštanskih brojeva. U drugom ciklusu od 80. do 100. naredbe vrši se ispitivanje da li je unet naziv grada u naredbi 70 u memoriji ili nije. Ako se nađe ime grada u memoriji, onda se izdaje odgovarajući izveštaj sa poštanskim brojem grada. Ako se ne nađe ime grada u memoriji, izdaje se izveštaj da ne postoji uneto ime grada u memoriji. Uočimo još da se imena gradova čuvaju u azbučnom nizu, a poštanski brojevi u odgovarajućim elementima brojnog niza. Izvršavanjem programa G743 može se dobiti sledeći protokol:

```
> RUN
NAZIV GRADA ?BEOGRAD
POŠTANSKI BROJ ?11000
NAZIV GRADA ?ZAGREB
POŠTANSKI BROJ ?41000
NAZIV GRADA ?LJUBLJANA
POŠTANSKI BROJ ?61000
NAZIV GRADA ?SARAJEVO
POŠTANSKI BROJ ?71000
NAZIV GRADA ?SKOPLJE
POŠTANSKI BROJ ?91000
NAZIV GRADA ?TITOGRAD
POŠTANSKI BROJ ?81000
NAZIV GRADA ?NOVI SAD
POŠTANSKI BROJ ?21000
NAZIV GRADA ?PRIŠTINA
POŠTANSKI BROJ ?38000
POŠTANSKI BROJ GRADA ?NOVI SAD
GRAD NOVI SAD IMA POŠTANSKI BROJ
21000
POŠTANSKI BROJ GRADA ?KRAGUJEVAC
U MEMORIJI NE POSTOJI GRAD SA IM
ENOM KRAGUJEVAC
POŠTANSKI BROJ GRADA ? —
```

Zahtev za izdavanje poštanskog broja za uneto ime grada na ulazu obrađuje se u beskonačnom programskom ciklusu

Zato se rad ovog programa može prekinuti samo pritiskom na taster BRK.

Kratak izvod

- Programski ciklusi mogu biti brojački i iterativni
- Ciklus koji ne sadrži izlazni kriterijum ponavlja se neograničen broj puta
- Naredba programskog ciklusa omogućuje programiranje ciklusa u obliku:

```
FOR i=v TO k [STEP p]
┌─────────────────────┐
│     telo ciklusa    │
└─────────────────────┘
NEXT i
```

gde v, k i p mogu biti brojni izrazi ali se ciklus izvršava samo za celobrojne vrednosti izraza
- Postoji samo jedan jednodimenzionalan brojni niz sa imenom A, koji se ne dimenzioniše, a moguć broj elemenata zavisi od raspoloživog memorijskog prostora
- Postoji samo jedan jednodimenzionalan azbučni niz sa imenom X$, koji se dimenzioniše naredbom

  ARR$(n)

  gde je n brojni izraz čija celobrojna vrednost određuje broj elemenata. Ovako uveden niz se uništava naredbom ARR$(—1)
- Kao indeks brojnog i azbučnog niza može se pisati brojni izraz, čija celobrojna vrednost određuje element niza. Vrednost indeksa može biti: 0, 1, 2, . .
- Vrednost azbučne indeksne promenljive može biti azbučni podatak maksimalne dužine 16.

Pitanja i zadaci za vežbu

1. Kakav je rezultat izvršavanja ciklusa

```
10 FOR I=32 TO 255
20 PRINT I, CHR$(I)
30 NEXT I
```

2. Sastaviti program koji izračunava vrednosti funkcije

$$y(x)=x+2.5 \cdot \mathrm{int}(x^2/3)$$

za $x=x_0+i \cdot \Delta x$, $i=0, 1, 2, \ldots, 10$

3. Zadat je niz među sobom različitih brojeva $x_1, x_2, \ldots, x_n$, $n \leqslant 20$. Sastaviti program koji određuje najmanji broj u zadatom nizu.
4. Zadat je niz među sobom različitih brojeva $x_1, x_2, \ldots, x_n$ $n \leqslant 10$. Sastaviti program koji određuje mesto najvećeg broja u zadatom nizu brojeva.

5. Sastaviti program koji omogućuje unošenje proizvoljnog broja dekadnih cifara, a zatim izdaje izveštaj u obliku

Cifra	Broj pojavljivanja
0	—
1	—
.	.
.	.
.	.
9	—

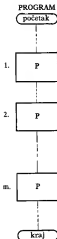
6. Sastaviti program koji izračunava sumu kvadrata parnih brojeva, počev od broja n do broja m, gde su brojevi n i m parni brojevi i treba ih uključiti u izračunavanje tražene mase.

7. Neka je $a_3\ a_2\ a_1\ a_0$ četvorocifren dekadni broj. Sastaviti program koji izračunava koliko postoji četvorocifrenih brojeva kod kojih je zbir prve dve cifre jednak zbiru sledeće dve cifre, tj. $a_3+a_2=a_1+a_0$

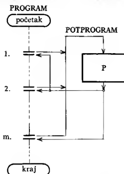
8
POTPROGRAMI

8.1. Organizacija programa

Često se prilikom rešavanja zadataka javlja potreba da se isti postupak ponovi na više mesta u programu. Na primer, korišćenje istih formula, ali za različite argumente, ili izvršavanje iste procedure, koju nije moguće uklopiti u ciklus. Program napisan za rešavanje takve vrste problema, sadržavao bi isti niz naredbi na različitim mestima programa. Celishodno bi bilo da se takav niz naredbi izdvoji u posebnu celinu koja se može koristiti po potrebi. Na takav način izdvojen niz naredbi naziva se *potprogram*. Obrazovanje potprograma omogućuje kraći zapis programa, samim tim



Sl. 8.1.1.



Sl. 8.1.2

i njegovo lakše prenošenje na računar i manje angažovanje memorijskog prostora u računaru.

Tako ako se niz od *n* naredbi označen sa P (sl. 8.1.1) u programu pojavljuje *m* puta, tada se ovaj niz može izdvojiti u posebnu celinu — potprogram koji se *m* puta poziva za izvršavanje (sl. 8.1.2). Program na sl. 8.1.1. pored ostalih naredbi sadrži *m.n* naredbi, jer se niz P od *n* naredbi ponavlja *m* puta. Ako se niz P od *n* naredbi izdvoji u potprogram (sl. 8.1.2), tada se u programu na mestima gde se nalazio niz P, vrši prelazak iz programa u potprogram. Po izvršenom potprogramu vrši se povratak u program i to neposredno iza mesta prelaska na potprogram. Ovde treba napomenuti da korišćenje potprograma u programiranju ne utiče na brže izvršavanje programa od strane računara. U primeru na sl. 8.1.2, umesto m.n naredbi, kako je dato na sl. 8.1.1, napisano je samo *n* naredbi, ali kada se izvršava program izvršiće se m.n naredbi. Ako bismo hteli da budemo potpuno precizni, onda je izvršavanje programa putem potprograma duže za trajanje naredbi prelaska iz programa u potprogram i povratka iz potprograma u program. Korišćenje potprograma u programiranju ima sledeća svojstva:

- pruža mogućnost kraćeg zapisa programa, a samim tim smanjuje mogućnost greške u pripremi programa,
- smanjuje angažovanje memorijskog prostora,
- omogućuje lakše testiranje programa, jer se potprogrami kao posebne programske celine mogu odvojeno testirati i
- isti potprogram može se koristiti u raznim programima.

U BASIC-jeziku, na računaru GALAKSIJA, može se koristiti jedna vrsta potprograma koji zovemo potprogramski segment.

8.2. Potprogramski segment

Potprogramski segment je niz označenih programskih redova koji se završava naredbom povratka u program. Prelazak iz programa na potprogramski segment vrši se naredbom

CALL n

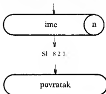
gde je n brojni izraz čija celobrojna vrednost određuje broj programskog reda na koji se vrši prelazak. Povratak iz potprogramskog segmenta u program vrši se naredbom

RETURN

pri čemu se u programu dolazi na naredbu koja sledi iza naredbe CALL kojom se prešlo na potprogramski segment. Broj potprogramskog reda n definiše mesto ulaza u potprogram, a naredba RETURN definiše mesto izlaza iz potprograma. Ova mesta se zovu ulazi, odnosno izlazi potpro-

grama. Potprogramski segment može imati više ulaza i više izlaza. U potprogramskom segmentu se ne navode posebne ulazne i izlazne veličine, jer se sve promenljive iz programa mogu koristiti i u potprogramu i obratno. Iz potprogramskog segmenta se ne može preći u drugi potprogramski segment.

U algoritamskim šemama prelazak na potprogramski segment označavaćemo grafičkim simbolom na sl. 8.2.1, gde je n broj koji označava ulaz u potprogram, a ime je simbolički naziv potprograma, koji se može koristiti u algoritamskim šemama, a ne javlja se u programu.



Sl. 8.2.1.

Sl. 8.2.2.

Oznaka ulaza n može biti ista kao i broj odgovarajućeg programskog reda u programskom segmentu. Povratak iz potprograma u program označavaćemo simbolom na sl. 8.2.2.

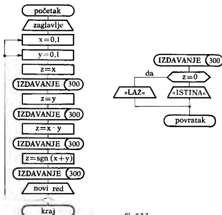
Primer.

Sastaviti program koji dodeljuje logičke vrednosti promenljivim x i y i izdaje tabelu vrednosti logičkih funkcija $x \wedge y$ i $x \vee y$.

Za predstavljanje logičkih konstanti u računaru koristićemo konstante 0 i 1, tako da 0 odgovara ⊥, a 1 logičkoj konstanti T. Onda se funkcija konjukcije može odrediti kao x y, a funkcija disjunkcije kao sgn (x+y), pri čemu je

$$\mathrm{sgn}(x+y) = \begin{cases} 1 \text{ ako je } x+y>0 \\ 0 \text{ ako je } x+y=0 \end{cases}$$

Za izdavanje tabele vrednosti logičkih funkcija uvešćemo da se 0 izdaje kao LAŽ, a 1 kao ISTINA. Algoritamska šema za ovaj zadatak prikazana je na sl. 8.2.3, gde je izdavanje tabele rešeno po koncepciji potprogramskog segmenta. Potprogramski segment je nazvan IZDAVANJE, a u BASIC-programu počeće programskim redom broj 300. Program se može zapisati u obliku:



Sl. 8.2.3.

```
10 !PROGRAM G821
20 PRINT"X          Y          X I Y   X ILI Y":PRINT
30 FOR X=0 TO 1
40 FOR Y=0 TO 1
50 Z=X:CALL 300
60 Z=Y:CALL 300
70 Z=X*Y:CALL 300
80 IF X+Y>0 Z=1
90 IF X+Y=0 Z=0
100 CALL 300
110 PRINT
120 NEXT Y
130 NEXT Y
140 STOP
300 IF Z=0 GOTO 330
310 PRINT"ISTINA ";
320 RETURN
330 PRINT"LAŽ       ";
340 RETURN
```

U ovom primeru potprogramski segment ima jedan ulaz (red 300) i dva izlaza (naredbe u redovima 320 i 340). Izvršavanjem programa G821 dobija se sledeći izveštaj:

```
> RUN
  X           Y           X I Y       X ILI Y
LAŽ         LAŽ         LAŽ         LAŽ
LAŽ         ISTINA      LAŽ         ISTINA
ISTINA      LAŽ         LAŽ         ISTINA
ISTINA      ISTINA      ISTINA      ISTINA
> _
```

8.3. *Rešeni zadaci*

8.3.1 Izračunavanje kvadratnog korena zapisati u obliku potprograma. Ovako zapisan potprogram iskoristiti za izračunavanje vrednosti funkcije

$$y(x) = \sqrt{x^2+4x+5{,}6}$$

u intervalu [0,5] sa korakom $\Delta x = 1$.
Rešenje: Program ne sadrži ulazne veličine, jer su u zadatku zadate vrednosti nezavisno promenljive za koje treba izračunati vrednosti funkcije. Program sadrži ciklus u kojem nezavisno promenljiva uzima vrednosti 0, 1, 2, 3, 4 i 5. Pošto su to celi brojevi to nezavisno promenljiva X može biti indeks ciklusa. Tako se program može zapisati u obliku:

```
I0 !PROGRAM G83I
20 PRINT" X"," Y(X)":PRINT
30 FOR X=0 TO 5
40 Y=X*X+4*X+5 6
50 CALL 600
60 PRINT X,Z
70 NEXT X
80 STOP
600 P=Y+1
610 Z=(P+Y/P)/2
620 IF P—Z<0 0005 RETURN
630 P=Z:GOTO 6I0
```

U potprogramu je predviđeno izračunavanje kvadratnog korena sa tačnošću 5×10^{-4}. Izvršavanjem programa dobija se sledeći izveštaj:

```
> RUN
X           Y(X)
0           2 36643
I           3 25576
2           4 19523
3           5 I5752
4           6 I3188
5           7.11337
> _
```

8.3.2 Sastaviti tri potprogramska segmenta sa sledećim funkcijama:

1. Vrši dodeljivanje vrednosti sa ulaza elementima brojnog niza i to od elementa A do elementa B.

2. Vrši sumiranje K parova elemenata brojnog niza i to od elementa A i od elementa B, a rezultat postavlja od elementa C.

3. Vrši izdavanje elemenata brojnog niza od elementa C, u obliku matrice od N vrsta i M kolona.

Sastaviti program koji koristi ova tri potprogramska segmenta za unošenje, sabiranje i izdavanje matrica.

Rešenje: Potprogramski segment za unošenje može se zapisatu u obliku:

```
300 FOR I=A TO B
310 INPUT A(I)
320 NEXT I
330 RETURN
```

Potprogramski segment za sabiranje elemenata niza i postavljanje rezultata u niz, može se zapisati u obliku:

```
400 FOR I=0 TO K—1
410 A(C+I)=A(A+I)+A(B+I)
420 NEXT I
430 RETURN
```

Treći potprogram vrši izdavanje elemenata brojnog niza u obliku matrice i to vrstu po vrstu matrice. Potprogram se može zapisati u obliku:

```
500 FOR I=1 TO N
510 FOR J=1 TO M
520 PRINT A(J+(I—1)*M+C—1),
530 NEXT J
540 PRINT
550 NEXT I
560 RETURN
```

Na kraju ćemo zapisati program koji vrši unošenje, sabiranje i izdavanje matrica pri čemu se pozivaju gore navedeni potprogrami:

```
10 'PROGRAM G832
20 PRINT"BROJ VRSTA ";:INPUT N
30 PRINT"BROJ KOLONA ";:INPUT M
40 PRINT"UNETI MATRICU A PO VRSTAMA·"
50 A=1:B=N*M:CALL 300
60 PRINT"UNETI MATRICU B PO VRSTAMA "
70 A=B+1·B=2*B·CALL 300
80 A=1 B=N*M+1:C=2*B—1.K=N*M.CALL 400
90 PRINT"MATRICA C=A+B "
100 CALL 500
110 STOP
```

Izvršićemo program G832 u cilju sabiranja matrica:

$$\begin{Bmatrix} 21 & 45 & 70 \\ 33 & 59 & 27 \end{Bmatrix} + \begin{Bmatrix} 19 & -5 & 44 \\ 77 & 92 & 50 \end{Bmatrix}$$

Dakle, matrice imaju po dve vrste i po tri kolone. Izvršavanjem programa G832 dobija se sledeći protokol

```
> RUN
BROJ VRSTA ?2
BROJ KOLONA ?3
UNETI MATRICU A PO VRSTAMA

?21
?45
?70
?33
?59
?27

UNETI MATRICU B PO VRSTAMA.
?19
?—5
?44
?77
?92
?50
MATRICA C=A+B:
40      40      114
110     151     77
> _
```

Kao što se vidi iz izveštaja, izračunata je matrica C u obliku:

$$\begin{Bmatrix} 40 & 40 & 114 \\ 110 & 151 & 77 \end{Bmatrix}$$

Kratak izvod

- Potprogram je niz naredbi programa koji se može neograničen broj puta pozivati iz programa
- Potprogramski segment je vrsta potprograma u BASIC-jeziku koji se poziva iz programa naredbom

  CALL (broj reda)

- Povratak iz potprogramskog segmenta u program vrši se naredbom

  RETURN

- Nije dozvoljeno korišćenje potprogramskih segmenata po dubini.

Pitanja i zadaci za vežbu

1. Objasniti zašto se naredbom GOTO ne može preći na potprogram.
2. Sastaviti potprogramski segment koji u zadatom delu brojnog niza određuje najmanji i najveći broj. Napisati program koji koristi navedeni potprogram.
3. Napisati potprogram za izračunavanje korena kvadratne jednačine

$$ax^2+bx+c=0$$

Sastaviti program koji koristi ovaj potprogram.
4. Sastaviti potprogram koji izračunava vrednosti funkcije

$$y(x)=\begin{cases} x+\sqrt{x-1} & \text{ako je } x>1 \\ x^2+4{,}5 & \text{ako je } x\leq 1 \end{cases}$$

Sastaviti program za testiranje ovog potprograma.
5. Zadata su dva prirodna broja x i y. Sastaviti program koji određuje najveći zajednički delilac ovih brojeva. Određivanje najvećeg zajedničkog delioca sastaviti u vidu potprogramskog segmenta.

9

RAD SA DATOTEKAMA

9.1. Organizacija datoteka

U programiranju se javljaju skupovi podataka, koji mogu biti predmet obrade jednog ili više programa, ili skupovi naredbi koji čine program. Ovako organizovani skupovi informacija (podataka ili naredbi) zovu se *teke*. Ako teka sadrži podatke zove se *datoteka*, a ako sadrži program zove se *programoteka*. Datoteke se najčešće organizuju na spoljnim memorijskim medijumima. Prenošenje podataka iz unutrašnje memorije u datoteku zove se *upis u datoteku*, a prenošenje podataka iz datoteke u unutrašnju memoriju zove se *čitanje datoteke*. Podskup podataka koji predstavlja logičku celinu sa gledišta korišćenja datoteke zove se *logički slog*. Podskup podataka koji se može preneti između unutrašnje memorije i spoljnje memorije i obratno, pri jednom obraćanju spoljnjem memorijskom medijumu zove se *fizički slog*.

Datoteka u koju se slogovi upisuju jedan za drugim, i iz koje se čitaju istim redosledom kojim su upisani zove se *sekvencijalna datoteka*. Ovakve datoteke se najčešće organizuju na neadresivim memorijskim medijumima, kakva je magnetna traka. Datoteka u kojoj se može upisati ili pročitati proizvoljan slog zove se *rasuta datoteka*. Ovakva organizacija datoteke se može primeniti samo na adresivim memorijskim medijumima kakav je disk. Skup podataka koji je sastavni deo programa i ne može biti korišćen od strane drugih programa već samo od onog u okviru kojeg se nalazi zove se *programska datoteka*. Na računaru GALAKSIJA može se koristiti samo programska datoteka.

9.2. Programska datoteka

Skup podataka organizovanih u jednu celinu u okviru programa čini programsku datoteku. Podaci u ovoj datoteci se definišu pomoću naredbe podataka:

⟨podatak⟩ [,⟨podatak⟩]*

Podatak može biti brojni i azbučni. Brojni podaci se mogu pisati u pozicionom ili eksponencijalnom obliku. Azbučni podaci se moraju pisati između znakova navoda i njihova dužina ne sme biti veća od 16. Unutar ovako zapisanog podatka može se pisati i zarez. Tako se može pisati:

34 5,—18,"NOVI SAD","BEOGRAD",158

gde je navedeno 5 podataka, od kojih su 1., 2. i 5. brojni podaci, a 3. i 4. azbučni podaci.

U jednom programu može biti veći broj naredbi podataka. Sve ove naredbe definišu jednu sekvencijalnu datoteku. U ovakvoj sekvencijalnoj datoteci, podaci slede u istom redosledu kao i naredbe podataka u programu. Podaci iz programske datoteke mogu se dodeljivati promenljivim u programu. Ovo se vrši pomoću posebne *naredbe čitanja* podataka iz programske datoteke koja se piše u obliku

TAKE(promenljiva)[,(promenljiva)]*

Čitanje podataka iz programske datoteke uvek se vrši sleva nadesno. Podatak iz programske datoteke i promenljiva u listi naredbe čitanja, kojoj se ovaj podatak dodeljuje, moraju se slagati po vrsti, što znači da se brojnoj promenljivoj može dodeliti samo brojni podatak, a azbučnoj promenljivoj azbučni podatak. Ukoliko se pokuša dodela brojnoj promenljivoj azbučnog podatka, tada se izdaje izveštaj o nastaloj grešci. Koji će podatak biti dodeljen promenljivoj iz liste naredbe čitanja definiše *pokazivač podatka* programske datoteke. Pokazivač podatka sadrži redni broj podatka koji je na redu za čitanje. Na početku rada programa (posle izvršavanja komande RUN) pokazivač podataka ukazuje na prvi podatak u programskoj datoteci. Posle svakog čitanja podatka iz datoteke pokazivač podatka uvećava svoju vrednost za jedan i tako ukazuje na sledeći podatak koji je na redu za čitanje. Ukoliko se zahteva čitanje podatka iz datoteke, a već su svi podaci pročitani, izdaje se izveštaj o nastaloj grešci. Pokazivač podatka programske datoteke može se programski, kada god se to želi u programu, vratiti na prvi podatak datoteke, bez obzira na njegov tekući sadržaj. Ovo je omogućeno naredbom za inicijalizaciju programske datoteke, koja se piše u obliku

TAKE 0

Ukoliko se želi postaviti pokazivač podataka na određenu naredbu podataka u programskoj datoteci, to se može učiniti naredbom

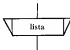
TAKE n

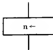
gde je n broj reda naredbe podataka od koje počinje čitanje podataka. Može se u naredbi TAKE koristiti i brojni izraz, ali prvi član ovog izraza mora biti brojni podatak. Tako se može pisati

TAKE 0+I

pri čemu se pokazivač podataka postavlja na broj reda zadat vrednošću promenljive I.
Programska datoteka formira se kada se formira i program. Za vreme izvršavanja programa mogu se samo čitati podaci iz ovakve datoteke, a ne može se vršiti upis u datoteku.

U algoritamskim šemama čitanje podatka označavaćemo simbolom na sl. 9.2.,1, gde lista sadrži imena promenljivih kojima se dodeljuju vrednosti iz programske datoteke.





Sl. 9.2.1. Sl. 9.2.2.

Postavljanje pokazivača na određenu naredbu podataka označićemo grafičkim simbolom na sl. 9.2.2, gde n ukazuje na broj reda naredbe podataka, a ako se pokazivač postavlja na početak programske datoteke, tada se može navesti broj reda, ali i ne mora, jer se ova situacija može označiti samo strelicom.

Primer.

Sastaviti program koji izračunava i izdaje vrednosti funkcije $y(x)$ za 5 zadatih vrednosti argumenta x.

Program će biti napisan tako da korisnik unosi funkciju, pre izvršavanja programa, u obliku jednog programskog reda

⟨broj reda⟩ Y=⟨brojni izraz⟩

gde je brojni izraz određen funkcijom $y(x)$. Pošto se vrednosti funkcije izračunavaju za fiksne vrednosti argumenta x, to je pogodno ove vrednosti čuvati u obliku programske datoteke podataka. Program se može zapisati u obliku:

```
10 !PROGRAM G921
20 PRINT" X"," Y(X)":PRINT
30 FOR I=1 TO 5
40 TAKE X
50 Y=X*X—2*X+3.5
60 PRINT X,Y
70 NEXT I
80 # 1,1.5,2.5,4,6
90 STOP
```

Funkcija, čije se vrednosti izračunavaju, definisana je u naredbi pod brojem reda 50. Prema tome, program izračunava vrednosti funkcije:

$$y(x)=x^2-2x+3,5$$

a argumenti funkcije su zadati u naredbi podataka pod brojem reda 80. Naredbom pod brojem reda 40, vrši se čitanje podataka iz programske datoteke, i to redom sleva nadesno (1, 1,5 itd). Izvršavanjem programa G921 dobija se sledeći izveštaj:

```
> RUN
X       Y(X)

1       2.5
1.5     2.75
2.5     4.75
4       11.5
6       27.5
> _
```

Ako se želi program G921 primeniti za izračunavanje vrednosti neke druge funkcije, a za iste vrednosti argumenata, tada pre izvršavanja programa treba uneti naredbu sa brojem 50 u kojoj se definiše funkcija. Tako, ako se žele izračunati vrednosti funkcije $y(x)=x^3-8x+2$, tada treba uneti naredbu

```
50 Y=X*X*X—8*X+2
```

i ponovo izvršiti program G921.

9.3. Rešeni zadaci

9.3.1. Zadatak 7.4.3. rešiti tako da se nazivi gradova i njihovi poštanski brojevi nalaze u programskoj datoteci. Ovo je svakako i bolje rešenje, koje više odgovara prirodi zadatka.

Rešenje: Program se može zapisati u obliku:

```
10 !PROGRAM G931
20 ARR$(7)
30 FOR I=0 TO 7
40 TAKE A(I),X$(I)
60 NEXT I:PRINT
70 PRINT"POSTANSKI BROJ GRADA ",:INPUT X$
80 FOR I=0 TO 7
90 IF EQ X$(I),X$ GOTO 130
100 NEXT I
110 PRINT"U MEMORIJI NE POSTOJI GRAD SA
    IMENOM ";X$
120 GOTO 70
130 PRINT"GRAD ";X$;" IMA POSTANSKI BROJ";
    A(I)
140 GOTO 70
150 #11000,"BEOGRAD"
160 #21000,"NOVI SAD"
170 #38000,"PRIŠTINA"
180 #41000,"ZAGREB"
190 #61000,"LJUBLJANA"
200 #71000,"SARAJEVO"
210 #81000,"TITOGRAD"
220 #91000,"SKOPLJE"
```

Programska datoteka se nalazi od broja reda 150 do 220.

U programskom ciklusu (naredbe 30 do 60) vrši se čitanje sadržaja programske datoteke i dodeljivanje vrednosti elementima brojnog i azbučnog niza. Ostali deo programa je zapisan na isti način kao i u programu G743. Izvršavanjem programa G931 dobija se isti izveštaj kao i pri izvršavanju programa G743.

9.3.2. Sastaviti program za učenje Morzeove azbuke.
Rešenje.

Ovim zadatkom ćemo ilustrovati jednu interesantnu primenu računara u obrazovanju. Program može izdavati informacije korisniku i na taj način vršiti određenu obuku korisnika. Takođe se mogu postavljati pitanja korisniku i vršiti provera kako je korisnik savladao materiju u kojoj je obučavan. Provera može uključiti i ocenjivanje korisnika. Osnovna prednost ovakve primene računara u obrazovanju jeste što se programom može predvideti redosled u izlaganju materije, kao i u postavljanju pitanja, saglasno sposobnostima svakog korisnika.

Mi ćemo sastaviti program koji može služiti za obuku u poznavanju Morzeovih kodova slova latinske azbuke i kojim se može vršiti provera stečenog znanja. U tabeli 9.3.1 prikazani su Morzeovi kodovi slova latinske azbuke.

Program ćemo sastaviti na sledeći način:

```
10 !PROGRAM G932
15 ARR$(51)
20 PRINT"                                   ";
30 PRINT"                                   ";
40 PRINT"        MORZEOVA AZBUKA            ";
50 PRINT"                                   ";
60 PRINT"                                   ";
80 PRINT"AKO UNESETE:"
90 PRINT"      SLOVO-PROGRAM IZDAJE MORZEOV"
92 PRINT"                KOD UNETOG SLOVA"
94 PRINT"             ?—PROGRAM VAM DOSTAVLJA"
96 PRINT"                PITANJE,"
98 PRINT"        KRAJ-KRAJ PROGRAMA"
120 PRINT"POČNIMO ..."
130 PRINT"VAŠ ZAHTEV ";:INPUTX$
150 Y$="?":IF EQ X$,Y$ GOTO 280
160 Y$="KRAJ":IF EQ X$,Y$ GOTO 240
168 TAKE 0
170 FOR I=0 TO 50 STEP 2
180 TAKE X$(I),X$(I+1)
190 IF EQ X$,X$(I) GOTO 260
200 NEXT I
210 TAKE 0
220 PRINT"NISTE DOBRO SHVATILI KAKO RADI OVAJ
    PROGRAM!"
230 GOTO 80
240 PRINT"DO SLEDEĆEG SUSRETA!"
250 STOP
260 PRINT"SLOVO ";X$;" JE ";X$(I+1)
270 GOTO 130
280 N=INT(1+26*RND)
```

```
300 TAKE 0
310 FOR I=1 TO N
320 TAKE X$(I),X$(I+1)
330 NEXT I
340 PRINT"KOJE JE OVO SLOVO ";X$(I)
350 PRINT"VAŠ ODGOVOR ";:INPUT X$
370 IF EQ X$(I—1),X$ GOTO 440
380 PRINT"NIJE TAČNO! PONOVO!"
390 PRINT"VAŠ ODGOVOR ";:INPUT X$
410 IF EQ X$(I—1),X$ GOTO 460
420 PRINT"NIJE TAČNO! TO JE ";X$(I—1)
430 GOTO 130
440 PRINT"ODLIČNO!"
450 GOTO 130
460 PRINT"DA, TO JE TAČNO!"
470 GOTO 130
480 # "A",".—","B",—...","C",—.—.","D","—.."
490 # "E",".","F","..—.","G","——.","H","...."
500 # "I","..","J",".———","K","—.—","L",".—.."
510 # "M","——","N","—.","O","———","P",".——."
520 # "Q","——.—","R",".—.","S","...","T","—"
530 # "U","..—","V","...—","W",".——","X","—..—"
540 # "Y","—.——","Z","——.."
```

1. Na početku program očekuje saopštenje korisnika. Korisnik može uneti:

● slovo, našta program izdaje Morzeov kôd unetog slova,

Slovo	Morzeov kôd	Slovo	Morzeov kôd
A	.—	N	—.
B	—...	O	———
C	—.—.	P	.——.
D	—..	Q	——.—
E	.	R	.—.
F	..—.	S	...
G	——.	T	—
H		U	..—
I	..	V	...—
J	.———	W	.——
K	—.—	X	—..—
L	.—..	Y	—.——
M	——	Z	——..

Tabela 9.3.1

● znak pitanja, našta program postavlja pitanje korisniku i to slučajnim izborom jednog Morzeovog kôda slova latinske azbuke. Ako korisnik ne odgovori tačno program mu omogućuje da ponovi odgovor. Ako i drugi put ne odgovori tačno tada program izdaje tačan odgovor. Ako korisnik prvi put da tačan odgovor dobija pohvalu, a ako tek drugi odgovor bude tačan, program izdaje poruku da je odgovor tačan,

● reč KRAJ, na šta program završava rad.

Na početku program izdaje informaciju korisniku o funkciji programa, a zatim informaciju o načinu korišćenja programa. Slova latinske azbuke i odgovarajući Morzeovi kôdovi obrazuju programsku datoteku. Izvršavanjem programa G932 može se dobiti sledeći protokol:

```
> RUN
```

```
MORZEOVA AZBUKA

AKO UNESETE:
 SLOVO—PROGRAM IZDAJE MORZEOV
       KOD UNETOG SLOVA,
     ?—PROGRAM VAM POSTAVLJA
       PITANJE,
  KRAJ—KRAJ PROGRAMA.
POČNIMO . . .
VAŠ ZAHTEV ?F
SLOVO F JE  . . — .
VAŠ ZAHTEV ??
KOJE JE OVO SLOVO — . . .
VAŠ ODGOVOR ?S
NIJE TAČNO! PONOVO!
VAŠ ODGOVOR ?B
DA, TO JE TAČNO!
VAŠ ZAHTEV ?KRAJ
DO SLEDEĆEG SUSRETA!
>_
```

Kratak izvod

- Skup podataka koji se organizuje u okviru jednog programa zove se programska datoteka. Ova datoteka se formira pomoću naredbi

  # ⟨podatak⟩ [,⟨podatak⟩]*

- Čitanje sadržaja programske datoteke vrši se naredbom

  TAKE ⟨promenljiva⟩ [,⟨promenljiva⟩]*

- Vraćanje pokazivača na početak programske datoteke vrši se naredbom

  TAKE 0

- Postavljanje pokazivača na određenu naredbu podataka, u okviru programske datoteke, vrši se naredbom

  TAKE ⟨broj reda⟩

  gde broj reda ukazuje na naredbu podataka na koju se postavlja pokazivač.

1. Navedite primere zadataka u kojima je korisno organizovati programsku datoteku podataka.

2. Sastaviti program za izračunavanje veličina Z_i po formuli

$$Z_i = \frac{A}{k_i} + B \cdot t_i, \quad i = 1, 2, \ldots, 6$$

pri čemu su A i B ulazne veličine, a k_i, t_i se nalaze u programskoj datoteci.

3. Od podataka u tabeli 9.3.2 obrazovati programsku datoteku. Sastaviti program koji određuje vrednost robe,

Šifra robe	Jedinična cena
A103	54.20
D235	72.30
T418	5.90
A218	34.05
B300	75.00

Tabela 9.3.2.

pri čemu se za svaku navedenu šifru količina robe unosi sa ulaza. Rezultate prikazati u obliku odgovarajuće tabele sa prigodnim zaglavljem. Na kraju izlazne tabele izdati ukupnu vrednost, kao sumu pojedinačnih vrednosti robe

10
EKRAN I GRAFIKA

U ovoj glavi ćemo upoznati neke mogućnosti BASIC-a za računar GALAKSIJA koje nisu standardni deo BASIC-jezika. Zapravo, ovde se radi o mogućnostima koje se često sreću na mikroračunarskim sistemima namenjenim za ličnu upotrebu. To su mogućnosti kojima se obezbeđuje lako programiranje raznih igara na računaru, ali koje vrlo efikasno mogu biti upotrebljene i u raznim drugim primenama.

10.1. Planiranje ekrana

Ekran je medijum na kojem korisnik posmatra sve o čemu komunicira sa računarom. Zato su vrlo važne sve mogućnosti koje se odnose na uređenje izveštaja koji se izdaju na ekranu. Za prikazivanje teksta na ekranu postoji raspoloživo 512 ćelija, raspoređenih u 16 redova sa 32 ćelije u svakom redu. Takođe smo videli da kada je 16 redova popunjeno, onda se sledeći red unosi kao 16., a prvi tome se gubi prvi red sa ekrana. Ovakvo pomeranje teksta na ekranu zvacemo *klizanje*. Međutim, nije teško zamisliti primere rada programa kada je poželjno da ne dolazi do klizanja na manjem ili većem delu ekrana. Sprečavanje kliza-

i \ j
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0
32
64
96
128
160
192
224
256
288
320
352
384
416
448
480
zamrznut deo ekrana
klizeći deo ekrana

Sl. 10.1.1. Zamrzavanje naredbom HOME 69

nja na jednom delu ekrana zvaćemo *zamrzavanje* slike na ekranu. Zamrzavanje se postiže naredbom

HOME (broj pozicije)

gde je broj pozicije ceo broj koji određuje poziciju na ekranu do koje su sve pozicije od početka ekrana zamrznute Tako, na primer, naredba

HOME 69

ima efekat da su pozicije na ekranu od 0 do 68 zamrznute, a ostale klizeće (sl. 10.1.1.). Naravno da će se najčešće ova naredba koristiti da se određen broj redova u gornjem delu ekrana zamrzne, da bi se planirao za grafičke potrebe. Međutim, kao što pokazuje primer na sl. 10 1.1. može se zamrznuti i deo jednog reda, ako je to potrebno. Za uređenje izlaznih izveštaja na ekranu često će biti potrebno izbrisati ceo ekran, tako da se prethodno izdavani izveštaji izbrišu i novi izveštaji učine čitljivijim. Ovo se može postići pomoću tastaturne komande SHIFT/DEL, koja briše ekran, ali ne utiče na plan zamrzavanja. Međutim, ako se želi izvršiti brisanje ekrana iz programa, tada se to može učiniti naredbom

HOME

koja briše ekran, ukida zamrzavanje i izdaje pokazivač pozicije ekrana u levom gornjem uglu ekrana.

U zamrznut deo ekrana može se normalno vršiti izdavanje kao i u klizeći deo naredbom PRINT, razlika je samo u tome, što, kada se napuni ekran i dođe do klizanja sadržaja na ekranu u zamrznutom delu ekrana sadržaj se neće menjati Ovo ćemo ilustrovati u sledećem primeru.

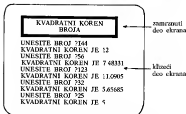
Primer.

Sastavićemo program za izračunavanje kvadratnog korena unetog broja, ali tako da na ekranu stalno stoji natpis KVADRATNI KOREN BROJA. Program se može zapisati u obliku:

```
10 !PROGRAM G10.1.1
20 HOME
30 PRINT"                              ";
40 PRINT"                              ";
50 PRINT"      KVADRATNI KOREN         ";
60 PRINT"          BROJA               ";
70 PRINT"                              ";
80 PRINT"                              ";
90 HOME 192
100 PRINT"UNESITE BROJ ",:INPUT X
110 Y=X+1
120 Z=(Y+X/Y)/2
130 IF Y—Z<0.0005 PRINT"KVADRATNI KOREN JE";
;Z:GOTO 100
140 Y=Z:GOTO 120
```

Napočetku programa izvršeno je brisanje ekrana, a zatim će se izdati uokviren natpis KVADRATNI KOREN BROJA

Ovaj natpis zauzima 6 redova, pa će se sa HOME I92 izvršiti zamrzavanje ovog dela ekrana. Unošenje brojeva i izračunavanje se vrši u beskonačnom ciklusu. Međutim, bez obzira koliko dugo se ciklus ponavljao rad programa će angažovati samo klizeći deo ekrana a to je od 7. do 16. reda. Izvršavanjem programa može se dobiti sledeći izveštaj na ekranu:



Uokvireni deo ekrana sa tekstom KVADRATNI KOREN BROJA je zamrznut a ostali deo je klizeći.

10.2. Unošenje znakova bez prikazivanja na ekranu

Nekada je pogodno da pritisak na taster ne proizvodi i izdavanje odgovarajućeg znaka na ekranu, jer to kvari postojeću sliku na ekranu. Za ovo postoji posebna funkcija koja se piše u obliku

KEY (0)

Vrednost ove funkcije je ASCII vrednost pritisnutog znaka na tastaturi. Ove vrednosti su prikazane u tabeli 5.7.I. Kada pri izvršavanju programa računar dođe do ove funkcije, tada se prekida izvršavanje i čeka se na pritisak bilo kojeg tastera na tastaturi. ASCII vrednost pritisnutog znaka uzima se kao vrednost funkcije i nastavlja se sa izvršavanjem programa. Pri ovome ne dolazi do izdavanja znaka na ekranu. Taster DEL daje vrednost 0, taster ENTER vrednost 13 a tasteri sa strehcama ↑, ↓, ←, → redom vrednosti 27, 28, 29 i 30. Tasteri BRK i STOP/LIST ne mogu se koristiti.

Primer:

Sastaviti program koji u desnom gornjem uglu ekrana izdaje pritisnut znak na tastaturi. Znak ćemo uneti funkcijom KEY (0). Program se može zapisati u obliku:

```
10 HOME
20 A=KEY (0)
30 PRINT AT 31, CHR$(A)
40 GOTO 20
```

Prvom naredbom programa izvrši se brisanje ekrana. U drugoj naredbi računar čeka na pritisak tastera na tastaturi. Kada se pritisne jedan taster, tada će ASCII vrednost odgovarajućeg znaka biti dodeljena brojnoj promenljivoj A i nastavlja se sa izvršavanjem naredbe koja sledi. U ovom slučaju dolazi do izdavanja unetog znaka u 31. ćeliji ekrana. Program, zatim, nastavlja sa ponavljanjem čekanja na pritisak tastera, njegovo izdavanje itd.

10.3. Gruba grafika

Gruba grafika se zasniva na korišćenju grafičkih simbola koji se mogu izdavati u okviru jedne ćelije na ekranu. Prostor jedne ćelije deli se na 6 *podćelija*, dve po horizontali i tri po vertikali. Kako po horizontali ima 32 ćelije to će biti 64 podćelije, a po vertikali ima 16 ćelija odnosno 48 podćelija. Tako se ceo ekran sastoji od $64 \times 48 = 3072$ podćelije. Osvetljen prostor jedne podćelije na ekranu zvaćemo *tačkom*. Naredba kojom se može osvetliti prostor bilo koje podćelije na ekranu, ili kako to programeri često kažu »upaliti tačka« ima sledeći izgled:

DOT x,y

gde su x i y brojni izrazi, čije celobrojne vrednosti određuju koordinate tačke na ekranu. Vrednost x mora biti iz intervala [0,63], a vrednost y iz intervala [0,47]. Tačka sa koordinatama (0, 0) nalazi se u levom gornjem uglu na ekranu (sl. 10.3.1). Na slici su prikazane adrese ćelija koje se koriste u konstrukciji AT u naredbi PRINT, kao i koordinate tačaka x i y u naredbi DOT. Posebna naredba omogućuje gašenje tačaka na ekranu. Ova naredba se piše u obliku

UNDOT x,y

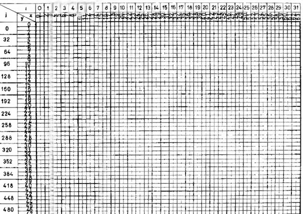
gde su x i y brojni izrazi, čiji celobrojni delovi određuju koordinate tačke na ekranu koja će biti ugašena ako je bila upaljena. Ako tačna nije bila upaljena naredba je bez dejstva. U programu je pogodno imati mogućnost ispitivanja da li je podćelija osvetljena ili nije. Ovo se postiže naredbom

IF DOT x,y(lista naredbi) [ELSE(lista naredbi)]

Kao što se vidi to je uobičajena IF naredba u kojoj je relacijski izraz DOT x,y. Ovaj relacijski izraz ima vrednost T ako je odgovarajuća podćelija na ekranu osvetljena, a vrednost ⊥ ako podćelija nije osvetljena. Veličine x i y mogu biti brojni izrazi čije celobrojne vrednosti definišu odgovarajuće koordinate tačke.

Primer. Sastaviti program koji na ekranu crta pravougaonik, tako da su osvetljene sve tačke na ivici ekrana. Program se može sastaviti na više načina, a jedno rešenje je sledeće:

Sl. 10.3.1. Planiranje ekrana AT(j+i), DOT x,y



```
10 HOME
20 FOR I=0 TO 63
30 DOT I, 0
40 DOT I, 47
50 NEXT I
60 FOR J=1 TO 46
70 DOT 0, J
80 DOT 63, J
90 NEXT J
95 GOTO 95
```

Na početku programa vrši se brisanje ekrana, a zatim u prvom ciklusu vrši crtanje horizontalnih duži, a u drugom ciklusu vertikalnih duži na ivicama ekrana. Horizontalne duži se crtaju sleva nadesno, a vertikalne odozgo nadole. Poslednja naredba, pod brojem 95, predstavlja ciklus u kojem se vrši stalno izvršavanje naredbe GOTO 95. Ovim se onemogućuje izdavanje poruke READY, što bi pokvarilo iscrtan okvir na ekranu. Ovakav kraj rada programa zvaćemo *dinamičko zaustavljanje* računara.

10.4. Časovnik

Računar GALAKSIJA raspolaže kvarcnim časovnikom koji omogućuje merenje vremena u stotim delovima sekunde, sekundama, minutima i časovima. Odbrojavanje časovnika vrši se u pedesetim delovima sekunde. Stanje časovnika se evidentira kao tekuća vrednost azbučne promenljive Y$. Zato se vrednost ove promenljive, pre startovanja časovnika, postavlja kao početna vrednost časovnika. Vrednost se mora postaviti u obliku

Y$="00:00:00:00"

gde sleva nadesno, po dve cifre, znače časove, minute, sekunde i stote delove sekunde. Jedinice vremena koje se ne žele meriti mogu se zdesna izostaviti. Tako, ako početno vreme postavimo

Y$="01:30:20"

to znači 1 čas, 30 minuta i 20 sekundi. Prema tome, u ovom slučaju neće se prikazivati stoti delovi sekunde. Na ovaj način smo postavili početno vreme na časovniku, a kada želimo da časovnik počne sa radom treba ga startovati naredbom:

DOT*

i sada časovnik radi. Ako želimo da pročitamo vreme na časovniku, to možemo učiniti naredbom

PRINT Y$

časovnik se zaustavlja naredbom

UNDOT*

Po zaustavljanju stanje časovnika je sačuvano kao vrednost azbučne promenljive Y$. Kada je časovnik jedanputa startovan on će raditi sve dok se ne zaustavi naredbom UNDOT*, bez obzira da li se program izvršava ili ne. Međutim, časovnik ne radi, iako je startovan, za vreme rada sa kasetofonom i, naravno, kada je računar isključen. Iz ovog razloga časovnik je pogodno koristiti kao štopericu koja

meri vreme između dva događaja u programu. Ovo nalazi primenu u raznim igrama na računaru, kada se meri vreme za koje se dostigne određen cilj u igri. Takođe se časovnik može korisno primeniti za ocenu brzine rada nekog programa i na taj način birati metode za realizaciju programa koje brže rade na računaru.

Primer.

Sastaviti program koji omogućuje merenje vremena izračunavanja kvadratnog korena po Njutnovoj formuli. Program se može sastaviti u obliku:

```
10 PRINT "UNESITE BROJ ";:INPUT X
20 Y$="00:00:00:00"
30 DOT*
40 Y=X+1
50 Z=(Y+X/Y)/2
60 IF Y—Z<0.0001 UNDOT*:GOTO 80
70 Y=Z:GOTO 50
80 PRINT"KVADRATNI KOREN ",Z
90 PRINT"VREME: ",Y$
100 GOTO 10
```

U naredbi 20 je postavljeno početno stanje časovnika. U ovom primeru časovnik počinje od nule. U 30. naredbi startovan je časovnik, a između broja reda 40 i 70 vrši se izračunavanje kvadratnog korena unetog broja X sa tačnošću 0.0001. Kada je postignuta ova tačnost vrši se zaustavljanje časovnika naredbom UNDOT* i prelazak na naredbu 80 u kojoj se izdaje vrednost izračunatog kvadratnog korena, a u naredbi 90 se izdaje vreme računanja. Izvršavanjem programa može se dobiti sledeći protokol:

```
> RUN
UNESITE BROJ ?2
KVADRATNI KOREN: 1.41421
VREME. 00:00:00:52
UNESITE BROJ ?9999999
KVADRATNI KOREN: 999.999
VREME. 00:00:01:54
UNESITE BROJ ?_
```

Kao što se vidi, kvadratni koren iz 2 izračunat je za 0.52 sec, a kvadratni koren iz 999999 za 1.54 sec. Rad prorama se može završiti tastaturnom komandom BRK.

10.5 Rešeni zadaci

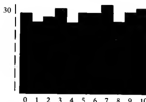
10.5.1. Sastaviti program koji crta histogram pseudoslučajnih brojeva iz intervala [0,10] generisanih funkcijom RND.

Rešenje: Program se može sastaviti u obliku:

```
10 !PROGRAM G10.5.1.
20 HOME
40 PRINT"HISTOGRAM SLUČAJNIH BROJEVA"
```

```
50 PRINT:PRINT"30!"
60 FOR I=1 TO 8.PRINT"     !":NEXT I
70 PRINT"  !—————————"——————"
80 PRINT"   0 1 2 3 4 5 6 7 8 9 10"
90 PRINT
95 FOR I=0 TO10·A(I)=NEXT I.N=0
100 A=INT(11*RND)·N=N+1
110 PRINT AT 448, "SLUČAJAN BROJ ";A
120 PRINT"BROJ GENERISANIH BROJEVA ",N
140 A(A)=A(A)+1
150 B=6+4*A
160 C=B+3
170 D=36—A(A)
180 E=35
190 FOR X=B TO C:FOR Y= D TO E:DOT X,Y
NEXT Y:NEXT X
200 IF A(A)=30 GOTO 200
210 GOTO 100
```

Program crta histogram sve dotle dok se jedan od brojeva između 0 i 10 ne javi 30 puta. Kada se ovo dogodi program se dinamički zaustavlja (naredba 200). Program izdaje generisan pseudoslučajni broj, kao i broj generisanih brojeva. Izvršavanjem programa G10. 5. 1 može se dobiti sledeća slika na ekranu



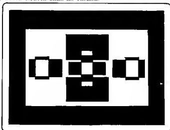
10.5 2. Sastaviti program koji crta pravougaonu površinu na ekranu, tako da se centar pravougaonika nalazi u

centru ekrana, a stranice pravougaonika da su paralelne sa ivicama ekrana. Koordinate jednog temena pravougaonika birati slučajnim izborom, a ostale odrediti simetrično u odnosu na centar ekrana. Unutrašnjost pravougaonika ispuniti tačkama, tako da tačka koja ne svetli bude osvetljena, a tačka koja svetli bude ugašena.

Rešenje: Program se može zapisati u obliku:

```
100 !PROGRAM G10 5.2.
110 HOME
120 X=INT(32*RND)
130 Y=INT(24*RNO)
140 FOR I=X TO 63—X
150 FOR J=Y TO 47—Y
160 IF OOT I, J UNDOT I, J:ELSE OOT I,J
170 NEXT J:NEXT I
180 FOR I=1 TO 2000
190 IF KEY(4) GOTO 220
200 NEXT I
210 GOTO 120
220 FOR X=0 TO 63
230 FOR Y=0 TO 47
240 IF OOT X,Y UNDOT X,Y ELSE OOT X,Y
250 NEXT Y:NEXT X
260 GOTO 260
```

Teme čije se koordinate slučajno biraju ima koordinate X∈[0,31] i Y∈[0,23]. U koncentričnim ciklusima od naredbe 140 do 170 vrši se prolazak kroz unutrašnjost pravougaonika, tako da se tačke pale ako ne svetle i gase ako svetle. Posle crtanja jednog pravougaonika program čeka 18 sekundi i ako korisnik pritisne taster 0 program prelazi na crtanje pravougaonika po ivici ekrana i vrši dinamičko zaustavljanje računara. Izvršavanjem programa G10.5 2 može se dobiti sledeća slika na ekranu.



Kratak izvod

- Zamrzavanje dela ekrana vrši se naredbom
  HOME (broj pozicije)
- Brisanje ekrana vrši se naredbom HOME
- Unošenje jednog znaka sa tastature, bez njegovog izdavanja na ekranu, vrši se funkcijom KEY(0)
- Osvetljenje podćelije sa koordinatama x i y vrši se naredbom

  DOT x,y

- Gašenje podćelije sa koordinatama x i y vrši se naredbom

  UNDOT x,y

- Ispitivanje da li je podćelija sa koordinatama x i y osvetljena ili ne vrši se naredbom
  IF DOT x,y(lista naredbi) [:(lista naredbi)]
- Početno stanje časovnika postavlja se kao vrednost azbučne promenljive Y$, kao na primer:

  Y$="00:00:00:00"

- Startovanje časovnika vrši se naredbom DOT*
- Zaustavljanje časovnika vrši se naredbom UNDOT*

Pitanja i zadaci za vežbu

1. Objasnite efekat zamrzavanja dela ekrana.
2. Zašto konstrukciju KEY(0) zovemo funkcijom, a ne naredbom ulaza?
3. Kakva je razlika između naredbi DOT i DOT*, kao i između naredbi UNDOT i UNDOT*
4. Sastaviti program koji unetu adresu za jedno lice izdaje u obliku adrese na poštanskom pismu okvirenu slučajno izabranim simbolima za okvir.
5. Sastaviti program po ugledu na zadatak 10.5.2, u kojem se neće crtati pravougaonik već samo izdavati jedna slučajno izabrana tačka i tri tačke koje odgovaraju temenima pravougaonika iz zadatka 10.5.2.
6. Sastaviti program za rešavanje tri jednačine sa tri nepoznate i izmeriti vreme rada programa koristeći naredbe za rad sa časovnikom.

11

BASIC-MAŠINSKI JEZIK

Naredbe BASIC-jezika se izvršavaju na računaru tako što se pomoću posebnog programa interpretiraju. Ovaj program je zapisan u ROM-memoriji na mašinskom jeziku. U mnogim primenama računara, ako se želi postići veća brzina rada programa, treba programe ili pojedine delove programa pisati na mašinskom jeziku, jer će se brže izvršavati na računaru od programa na BASIC-jeziku. Zato ćemo u ovoj glavi upoznati neke mogućnosti BASIC-jezika pomoću kojih korisnik može koristiti mašinski nivo računara. Naravno, da bi se mašinski nivo dobro iskoristio od strane programera treba znati mašinski jezik, u ovom slučaju, mikroprocesora Z80A. Ovde nećemo ulaziti u detalje programiranja na mašinskom jeziku, jer to nije predmet ove knjige, ali ćemo pokazati samo mogućnosti koje pruža BASIC-jezik za vezu sa mašinskim jezikom.

11.1. Heksadekadni brojevi

Konstrukcije mašinskog jezika su zapisane u binarnoj azbuci. Kako slova binarne azbuke možemo označiti sa 0 i 1, to su sve ove konstrukcije niske nula i jedinica. Ovakve niske nula i jedinica nisu pogodne za korišćenje od strane korisnika računara, pa se zato uvodi binarno kodirani heksadekadni brojni sistem u kojem se četiri binarne cifre čitaju kao jedna heksadekadna cifra. Prema tome, sadržaj jednog bajta u memoriji može se pročitati kao dve heksadekadne cifre, odnosno sadržaj dva bajta kao 4 heksadekadne cifre. U BASIC-u je dozvoljeno korišćenje heksadekadnih brojnih podataka. *Heksadekadni brojni podatak* je najviše četvorocifreni heksadekadni broj. Ovakav broj se u memoriji registruje u dva bajta u binarnom brojnom sistemu u potpunom komplementu. Heksadekadne cifre su 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E i F. Da bi se zapis heksadekadnog broja razlikovao od dekadnog ispred heksadekadnog broja piše se poseban specijalni znak (&). Tako je zapis &2AF1 heksadekadni broj 2AF1. Ako nas interesuje dekadna vrednost koja odgovara ovom heksadekadnom broju, to će biti:

$$(2AF1)_{16}=2\cdot 16^3+A\cdot 16^2+F\cdot 16^1+1\cdot 16^0=2\cdot 16^3+10\cdot 16^2+$$
$$+15\cdot 16^1+1\cdot 16^0=(10993)_{10}$$

Heksadekadni broj 2AF1 u memoriji računara se registruje u obliku niske nula i jedinica: 0010 1010 1111 0001. Očigledno da heksadekadni zapis brojeva omogućuje udobniji rad sa binarnim sadržajima, nego što je to sam binarni sadržaj ili dekadni brojni sistem.

Primer. Naredba

```
PRINT &FF+&FFFF
```

izdaje rezultat 254, jer je $(FF)_{16}=(255)_{10}$, a $(FFFF)_{16}=(-1)_{10}$

11.2. Slobodan memorijski prostor

Korisnik računara ima određen prostor u RAM-memoriji za svoj program. Ovaj prostor popunjava naredbama BASIC-jezika, podacima ili, kao što ćemo videti, potprogramima na mašinskom jeziku. Naravno, biće pogodno da korisnik može da dobije informaciju o slobodnom memorijskom prostoru da bi mogao da planira korišćenje ovog prostora. Ovo se može postići funkcijom MEM. Ova funkcija, koja nema argumenata, ima vrednost slobodnog memorijskog prostora u bajtovima. Na primer, naredba

PRINT MEM

može izdati broj 5062 što znači da u RAM-memoriji postoji slobodnih 5062 bajta za korisnika

11.3. Funkcija čitanja sadržaja memorije

Posebnim funkcijama omogućeno je čitanje sadržaja memorijskog podregistra ili registra. U engleskoj literaturi je uobičajeno da se sadržaj podregistra (8 ćelija) zove *bajt*, a sadržaj registra (16 ćelija) zove se *reč*. Funkcija kojom se čita vrednost jednog bajta, piše se u obliku:

BYTE(a)

gde je *a* brojni izraz čija celobrojna vrednost predstavlja adresu podregistra u memoriji čiji se sadržaj javlja kao vrednost funkcije. Sadržaj se tumači kao osmobitni binarni neoznačen broj. Prema tome, vrednost funkcije biće iz intervala [0,255].

Funkcija kojom se čita vrednost jedne reči, piše se u obliku

WORD(a)

gde je *a* brojni izraz čija celobrojna vrednost predstavlja adresu registra u memoriji čiji se sadržaj javlja kao vrednost funkcije. Sadržaj se tumači kao 16-to bitni binarni označen broj iz intervala $[-2^{15}, 2^{15}-1]$, što u dekadnom obliku ima vrednost [—32768,32767]

Primer. Sledeći program

```
10 X$="NIS"
20 A=PTR X$
30 PRINT BYTE(A), BYTE(A+1), BYTE(A+2)
40 STOP
```

dodeljuje azbučnoj promenljivoj X$ azbučni podatak NIS, a brojnoj promenljivoj A adresu polja u memoriji u kojem se nalazi tekuća vrednost promenljive X$. U naredbi broj 30 vrši se izdavanje bajtova iz polja rezervisanog za vred-

nost promenljive X$ i to prva tri bajta. U našem primeru računar će izdati vrednosti

78 73 94

a to su dekadne vrednosti kôdnih reči u ASCII-kôdu redom za slova N, I i Š (vidi tabelu 5.7.1).

11.4. Naredba za upis u memoriju

Ako želimo da menjamo sadržaj memorijskih podregistara i registara ili da upisujemo novi sadržaj u njih, to možemo ostvariti naredbama za upis u memoriju. Tako naredba

BYTE a,s

gde *a* i *s* mogu biti brojni izrazi, upisuje sadržaj *s* u memorijski podregistar sa adresom *a*. Sadržaj koji se upisuje može biti najviše dvocifren heksadekadni broj koji se u memorijski podregistar upisuje kao 8-mo bitni binarni broj.

Na sličan način se može upisati sadržaj u memorijski registar sa naredbom

WORD a,s

gde *a* i *s* mogu biti brojni izrazi. U ovom slučaju se sadržaj *s* upisuje u memorijski registar sa adresom *a*. Sadržaj koji se upisuje može biti najviše četvorocifreni heksadekadni broj koji se u memorijski registar upisuje kao 16-to bitni binarni broj.

Primer 1. Sledeći program

```
10 X$="NOVI BEOGRAD"
20 FOR I=0 TO 3
30 BYTE PTR X$+I, 32
40 NEXT I
50 PRINT X$
60 STOP
```

dodelju tekst NOVI BEOGRAD azbučnoj promenljivoj X$, zatim se u ciklusu vrši brisanje reči NOVI postavljanjem praznina (kod 32, vidi tabelu 5.7.1). Na kraju, program izdaje vrednost azbučne promenljive X$ i to će biti reč BEOgrad, a ispred reči se izdaje 5 praznina.

Primer 2. Sledeći program

```
10 A=31
20 WORD PTR A+1, &7800
30 PRINT A
40 STOP
```

dodeljuje promenljivoj A vrednost 31. U naredbi 20 menja se ova vrednost, tako, kada se izda vrednost promenljive A

biće to broj 15 5. Za razumevanje ovog zadatka neophodno je poznavanje načina registrovanja brojeva u pokretnom zarezu u memoriji računara. Kako to nije predmet ove knjige, to nećemo objašnjavati ovaj zadatak dublje, već ostavljamo čitaocima koji poznaju prikazivanje brojeva u pokretnom zarezu da objasne kako je dobijen broj 15.5 u ovom programu. Objašnjenje načina registrovanja brojeva u pokretnom zarezu može se naći u knjigama u kojima se izučava mašinski jezik [2].

11.5. Potprogrami na mašinskom jeziku

U ovom odeljku ćemo videti da korisnik može napisati i iz BASIC-programa pozvati potprogram na mašinskom jeziku. Međutim, javlja se problem u kojem delu memorije registrovati potprogram na mašinskom jeziku, a da se ovaj ne preklopi sa zonom BASIC-programa ili nekim podacima koji se koriste u programu. Zato ćemo morati da upoznamo raspored programa i podataka u memoriji. Ovo je prikazano u tabeli 11.5.1. U tabeli je prikazana adresa u heksadekadnom i dekadnom obliku, kao i namena odgovarajućeg memorijskog prostora sa oznakom vrste memorije. Kao što se vidi iz tabele slobodan prostor se može naći samo između zone programa i brojnog niza. Međutim, zona programa se širi prema većim adresama, a brojni niz prema manjim. Iz ovog razloga ovaj deo memorije nije pogodno koristiti za potprograme na mašinskom jeziku. Zato BASIC-interpretator raspolaže komandom kojom se početak zone programa može pomeriti prema višim adresama. To je komanda

NEW n

gde je n dekadni broj koji određuje broj bajtova za koje se pomera početak zone programa. Tako će novi početak zone programa biti od adrese 11322+n. Potprogram mora biti napisan na mašinskom jeziku mikroprocesora Z80A. Ovakav potprogram može se pozvati iz BASIC-programa funkcijom

USR(a)

gde je *a* adresa prve naredbe potprograma na mašinskom jeziku. Vrednost potprograma USR je sadržaj registra HL u mikroprocesoru Z80A. Ova vrednost ne mora da bude značajna kao rezultat rada potprograma, jer potprogram može da vrši značajnu obradu podataka, a da se to ne vidi kroz sadržaj registra HL. S obzirom da potprogram USR ima ulogu funkcije to se poziva u programu tako što se može navesti kao argument brojnog izraza. Najčešće će biti pogodno prelazak na potprogram izvršiti naredbom

Y=USR(a)

gde dodeljena vrednost promenljivoj Y ne mora biti od značaja.

Adresa		Namena adresnog prostora	Vrsta memorije
heksade-kadna	dekadna		
0000 . . . 0FFF	0 . . . 4095	Sistemski program BASIC-interpretator	ROM
1000 . . . 1FFF	4096 . . . 8191	Za proširenja	ROM
2000 . . . 27FF	8192 . . . 10239	Koristi sistem	RAM
2800 . . . 29FF	10240 . . . 10751	Slika na ekranu	RAM
2A00 . . . 2C39	10752 . . . 11321	Promenljive i pokazivači	RAM
2C3A . . . 37FF	11322 . . . 14335	Zona programa ↓ Slobodan prostor Brojni niz ↑ Azbučni niz ↑	RAM

Tabela 11.5.1 Namena memorijskog prostora

Korisnici koji žele da koriste mašinski nivo programiranja mogu korisnim intervencijama na sadržaj registara, koji su inače pod kontrolom sistemskih programa, proizvesti korisne i interesantne akcije u svom programu. Za to je neophodno dobro poznavanje funkcije pojedinih sistemskih potprograma i programa. Korisnik može pogrešnim intervencijama onemogućiti ispravan rad sistemskih programa, ali ne može ovim intervencijama pokvariti računar. Ako u jednom trenutku korisnik primeti da računar normalno ne funkcioniše, treba pritisnuti taster RESET

na zadnjoj ploči računara. Ako ovo ne pomogne, tada jednostavnim isključenjem i ponovnim uključenjem računara sve će biti dovedeno u normalno početno stanje. Međutim, da bi korisnik mogao da interveniše na nivou sistemskih programa mora da poznaje adrese značajnih registara i zona u memoriji i njihovu namenu u sistemskim programima. U tabeli 11.5.2 prikazane su neke od ovih adresa. U sledećim primerima ilustrovaćemo korišćenje sistemskih potprograma i pisanje potprograma na mašinskom jeziku.

Adresa zone		Veličina zone u bajtovima	Početno stanje	Namena zone
heksadekadna	dekadna			
2800	10240	512	20	slika na ekranu
2A00	10752	104	00	zona brojnih promenljivih (A—Z)
2A68	10856	2	2800	pozicija kursora
2A6A	10858	2	3800	kraj memorije
2A6C	10860	2	00	zamrzavanje ekrana
2A70	10864	16	00	promenljiva X$
2A80	10880	16	00	promenljiva Y$
2A95	10901	2	00	pokazivač broja reda u programu za vreme izvršavanja CALL i FOR-NEXT
2A99	10905	2	00	veličina azbučnog niza
2A9D	10909	2	2C3C	pokaizvač programske datoteke
2BA8	11176	1	0B	horizontalna pozicija slike na ekranu
2BAF	11183	1	00	startovanje časovnika (sedmi bit postavljen u jedinicu)
2BB0	11184	1	00	pomeranje slike (brojač)
2BB1	11185	1	00	pokazivač pomeranja slike
2BB6	11190	125	00	ulazni prihvatnik
2C36	11318	2	2C3A	pokaizvač početka BASIC-programa
2C38	11320	2	2C3A	pokazivač kraja BASIC-programa
2C3A	11322		00	zona BASIC-programa

Tabela 11.5.2 Adrese nekih zona sistemske namene

Primer 1.

Za vreme rada računar GALAKSIJA veliki deo vremena troši na održavanje slike na ekranu. Zapravo, 50 puta u sekundi proizvodi prekid izvršavanja programa da bi se opslužio ekran. Samo tako se može dobiti solidan kvalitet slike na ekranu. Ako korisnik želi da računar ne prekida proces računanja, radi održavanja slike na ekranu, to može postići onemogućavanjem prekida radi održavanja slike. Ovo se može postići pozivom potprograma USR(14), a ponovno omogućavanje prekida postiže se potprogramom USR(22) ili dolaskom na naredbu kraja programa STOP, kao i pritiskom na taster BRK ili RESET. Da bismo ilustrovali razliku u brzini rada računara bez održavanja i sa održavanjem slike na ekranu posmatrajmo sledeći program:

```
10 X=USR(14)
20 Y=0
30 FOR I=1 TO 2000
40 Y=Y+I*I
50 NEXT I
60 PRINT Y
70 STOP
```

Ako se ovaj program izvrši komandom RUN, tada će na početku programa (naredba 10) biti onemogućen prekid programa, neće se održavati slika na ekranu i program će raditi oko 18 sekundi. Međutim, ako isti program izvršimo sa RUN 20 tada neće doći do onemogućavanja prekida, i program će se izvršavati sa održavanjem slika na ekranu. U ovom slučaju program će trajati oko 53 sekunde.

Primer 2.

Sadržaj podregistra sa adresom 2BA8 odgovoran je za horizontalni položaj slike na ekranu. Početno stanje ovog podregistra je $(0B)_{16}$. Međutim, ako se ova vrednost promeni, na primer, naredbom

BYTE &2BA8,&F

na &F, tada će doći do pomeranja slike na ekranu udesno. Vraćanje slike može se postići ponovnim postavljanjem $(0B)_{16}$ u podregistar 2BA8.

11.6. Snimanje mašinskih programa

Komandom SAVE snima se sadržaj zone programa na kasetofonu. Međutim, ako se želi snimiti bilo koji deo memorije tada se može koristiti komanda

SAVE n,m

gde su n i m početna i krajnja adresa zone u memoriji čiji se sadržaj snima. Unošenje ovako snimljenog programa sa kasete u RAM-memoriju vrši se komandom OLD. Međutim, ako je napisan poziciono-nezavisan potprogram na mašinskom jeziku, onda se ovakav potprogram može uneti sa kasete u RAM-memoriju komandom:

OLD k

gde je k dekadni broj koji određuje broj bajtova za koji se pomera upis u memoriji u odnosu na početak zone programa (&2C3A). Za k>0 pomeranje se vrši prema višim adresama, a za k<0 prema nižim adresama u memoriji.

Ako se za mašinski potprogram rezerviše memorijski prostor pomeranjem zone programa (komanda NEW n), tada se snimanje mašinskog potprograma i BASIC-programa može izvršiti komandom SAVE, jer ova naredba snima deo memorije od adrese &2C3A do kraja BASIC-programa. Čitanje ovako snimljenog programa vrši se naredbom OLD. Verifikacija izvršenog snimanja može se izvršiti komandom OLD?.

11.7.1. Sastaviti BASIC-program koji upisuje u memoriju potprogram na mašinskom jeziku koji postavlja uneti znak sa ulaza u sve ćelije ekrana.
Rešenje: Potprogram na mašinskom jeziku biće zapisan u okviru BASIC-programa u obliku programske datoteke. Pre unošenja programa treba primeniti komandu NEW 15, da bi se oslobodio prostor za smeštaj mašinskog potprograma. Program se može zapisati u obliku

```
10 FOR I=&2C3A TO &2C48
20 TAKE A
30 BYTE I,A
40 NEXT I
50 INPUT X$
60 BYTE &2C41,BYTE(PTR X$)
70 X=USR(&2C3A)
80 GOTO 80
90 # &01,&00,&02,&21,&00,&28,&36,&41
100 # &23,&0B,&CB,&78,&28,&F8,&C9
```

Izvršavanjem programa vrši se upis mašinskog potprograma od adrese 2C3A do 2C48. Potprogram je zapisan u vidu heksadekadnih brojeva u naredbama pod brojevima 90 i 100. Unošenjem bilo kojeg znaka sa tastature, izvrši se postavljanje odgovarajućeg kôda u potprogram (adresa 2C41), a zatim prelaskom na potprogram (naredba 70) izvrši se mašinski potprogram od adrese 2C3A. Poslednja naredba potprograma (C9) izvrši povratak u program u kojem dolazi do dinamičkog zaustavljanja računara.

11.7.2. Sastaviti program koji promenljivim A,B,C, ..., Z dodeljuje redom brojeve 1, 2, 3, ..., 26.
Rešenje: Primenićemo malo neobično rešenje, takvo, da se u programu izvrši zahtevano postavljanje primenom programske modifikacije. Program se može zapisati u obliku

```
10 FOR I=1 TO 26
20 BYTE 11358,64+I
30 A=I
40 NEXT I
50 I=9
60 STOP
```

Program se izvršava tako da se u naredbi 30 promenljiva A menja redom u promenljive A, B, C, ..., Z. Po izlasku iz ciklusa izvrši se postavljanje promenljive I na vrednost 9, jer je ova promenljiva indeks ciklusa, pa se njena vrednost menja u ciklusu. Posle izvršavanja ovog programa, korisnik se može uveriti u korektnost izdavanjem vrednosti promenljivih. Tako, na primer, naredba PRINT Z izdaje vrednost 26 itd.

U programiranju nije preporučljivo koristiti programsku modifikaciju, jer ova može biti uzrok grešaka koje se

vrlo teško otkrivaju u programu. Ovim zadatkom smo želeli samo da pokažemo kakvi neobični efekti mogu biti postignuti na nivou mašinskog jezika.

Kratak izvod

- Heksadekadni brojevi se pišu u obliku &nnnn, gde je n heksadekadna cifra.
- Slobodan memorijski prostor se može odrediti kao vrednost funkcije MEM.
- Sadržaj memorijskog podregistra može se dobiti kao vrednost funkcije

BYTE(a)

gde je *a* adresa podregistra.
- Sadržaj memorijskog registra može se dobiti kao vrednost funkcije

WORD(a)

gde je *a* adresa registra.
- Naredba za upis u podregistar

BYTE a,s

gde je *a* adresa podregistra, a *s* vrednost koja se upisuje.
- Naredba za upis u registar

WORD a,s

gde je *a* adresa registra, a *s* vrednost koja se upisuje.
- Poziv potprograma na mašinskom jeziku piše se u obliku funkcije

USR(a)

gde je *a* adresa prve naredbe potprograma.
- Pomeranje zone programa može se izvršiti komandom

NEW n

gde je n dekadni broj koji određuje broj bajtova za koji se vrši pomeranje.
- Snimanje delova memorije može se vršiti komandom

SAVE n,m

gde su n i m, početna i krajnja adresa zone u memoriji.
- Pomeranje adrese od koje se vrši upis sa kasete u memoriju, može se izvršiti komandom

OLD k

gde je k dekadni broj koji određuje broj bajtova za koji se vrši pomeranje

Pitanja i zadaci za vežbu

1. Objasniti kako se može efekat komande NEW 20 postići pomoću naredbi BASIC-jezika, promenom stanja registara koje koriste sistemski programi.

2. Uneti BASIC-program

```
10 INPUT X
20 PRINT X,X*X,X*X*X
30 STOP
```

i odrediti koliko bajtova zauzima u memoriji, kao i broj slobodnih bajtova u memoriji.

3. Objasniti kako se može postići veća brzina izvršavanja programa. Da li se za merenje vremena izvršavanja programa u režimu brzog rada može koristiti časovnik u računaru?

4. Koristeći funkciju BYTE ispitati način registrovanja BASIC-programa, iz zadatka 2, u zoni programa u memoriji.

12
PROGRAMSKE TEHNIKE

U prethodnim poglavljima upoznali smo BASIC-jezik za računar GALAKSIJU. Međutim, poznavanje jezika nije dovoljno za pisanje efikasnih programa. Za ovo je neophodna praksa. Kroz praksu se stiče iskustvo o pravilnoj i efikasnoj upotrebi konstrukcija jezika. Postupci koji mogu doprineti efikasnijem programiranju zovu se *programske tehnike*. Ove se dele na opšte i specijalizovane. *Opšte programske tehnike* nisu zavisne od jezika i konfiguracije sistema, dok se *specijalizovane* odnose na određen programski jezik ili konfiguraciju sistema. Ovde će biti reči o specijalizovanim programskim tehnikama za računar GALAKSIJU.

12.1. Grananje po vrednosti logičkog izraza

Upoznali smo naredbu grananja (IF)po vrednosti relacijskog izraza. Za ispitivanje složenih uslova može se koristiti veći broj naredbi uslovnog prelaska. Tako na primer

```
IF A<0 IF B>0 PRINT"BEOGRAD"
```

izdaje tekst BEOGRAD samo ako je A<0 i B>0. Ovako zapisane IF-naredbe mogu koristiti samo jednu ELSE opciju. Tako, programski red

```
IF A<0 IF B>0 PRINT"BEOGRAD":ELSE PRINT
"ZAGREB"
```

izdaje tekst BEOGRAD samo ako je A<0 i B>0, ali u svakom drugom slučaju, ako jedan od uslova ili oba nisu zadovoljeni, izdaje tekst ZAGREB. Obrazovanje složenih ispitivanja navođenjem više IF-naredbi može se uprostiti ako se iskoristi jedna jednostavna činjenica iz realizacije vrednosti relacijskog izraza u BASIC-interpretatoru. Već smo naveli da vrednost relacijskog izraza može biti ⊤, ako je navedena relacija tačna, odnosno ⊥ ako navedena relacija nije tačna. Ako se ima u vidu da se vrednost ⊤ ostvaruje kao brojna vrednost različita od nule, a vrednost ⊥ kao nula, onda se ovo može iskoristiti za rad sa logičkim izrazima. Tako, ako logičkoj vrednosti ⊤ pridružimo brojnu vrednost 1, a logičkoj vrednosti ⊥ brojnu vrednost nula, onda se ovo može iskoristiti za obrazovanje logičkih izraza u kojima se logičke operacije ∧ i ∨ zamenjuju aritmetičkim operacijama množenja i sabiranja (tabela I2.I.1). Argumenti ovakvog logičkog izraza mogu biti relacijski izrazi. Kao što se vidi iz table vrednost logičkih operacija biće 0, za ⊥, odnosno 1 za ⊤, pri čemu je izuzetak logička ILI operacija kada su oba argumenta jednaka jedinici. U ovom slučaju vrednost logičke operacije biće brojna vrednost 2. Me-

x	y	x · y	x+y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	2

Tabela 12.1.1. Ostvarenje logičkih operacija

đutim, računar će ipak korektno da tumači vrednost logičkog izraza, iz razloga što se u IF-naredbi brojna vrednost različita od nule tumači kao ⊤, a brojna vrednost nula kao ⊥. Imajući ovo u vidu argumenti logičkog izraza mogu biti aritmetički i relacijski izrazi. Ovde je jedini izuzetak slučaj

IF X+Y PRINT"BEOGRAD"

U ovom primeru, ako je X=p, a Y=—p gde je p≠0, tada je X+Y=0, što ne odgovara logičkoj interpretaciji za X=⊤ i Y=⊤, jer mora biti XVY=⊤. Ovo ćemo pokazati na sledećim primerima.

Primer 1.

Primer naveden na početku ovog odeljka

IF A<0 IF B>0 PRINT"BEOGRAD"

može se zapisati u obliku

IF(A<0)*(B>0) PRINT"BEOGRAD"

Ovaj zapis se zasniva na činjenici da, ako je A<0, tada relacijski izraz A<0 ima vrednost 1, a ako je B>0, tada relacijski izraz B>0 ima takođe vrednost 1, pa je proizvod ovih jedinica takođe jedinica, a to znači da su oba uslova zadovoljena.

Ovde treba napomenuti da je čitljivost drugog zapisa bolja ako se zvezdica (*) čita kao logička I operacija (∧). Međutim, nedostatak drugog zapisa je u tome što zvezdica označava operaciju množenja, pa to može dovesti do konfuzije, a i to, što se javlja operacija množenja, čini da se drugi zapis sporije izvršava na računaru.

Primer 2

Programski red

IF R=0 PRINT"R JE 0":ELSE PRINT"R NIJE 0"

izdaje tekst R JE 0, ako je R=0, odnosno R NIJE 0, ako je R≠0. Ovo se može zapisati i u obliku

IF R PRINT"R NIJE 0" ELSE PRINT"R JE 0"

Oba programska reda izdaju iste rezultate, a imaju različite zapise. Naravno, prvi zapis je bolji sa gledišta čitljivosti programa.

Primer 3. Sastaviti program koji za unete brojeve A i B sa ulaza izdaje tekst USLOV JE ISPUNJEN, ako je A<0 ili B>0. Program se može zapisati u obliku:

```
10 INPUT A:INPUT B
20 IF A<0 PRINT"USLOV JE ISPUNJEN":GOTO 10
30 IF B>0 PRINT"USLOV JE ISPUNJEN":GOTO 10
40 GOTO
```

Međutim, ako koristimo mogućnosti zapisa logičke operacije ILI, tada se gornji program može zapisati u obliku

```
10 INPUT A:INPUT B
20 IF(A<0)+(B>0) PRINT"USLOV JE ISPUNJEN"
   :GOTO 10
30 GOTO 10
```

U slučaju kada su oba uslova A<0 i B>0 tačna, izračunava se vrednost izraza (A<0)+(B>0) kao brojna vrednost 2, a to se uzima kao tačna vrednost logičkog izraza u IF-naredbi.

12.2. Rad sa azbučnim veličinama

U BASIC-u za računar GALAKSIJU može se ispitivati samo jednakost azbučnih promenljivih. Ovo nije dovoljno za uređenje azbučnih podataka u azbučni redosled, a takođe ne postoji ni veći broj azbučnih funkcija. Ovo se može nadoknaditi izgradnjom potprograma koji obavljaju funkcije za koje nedostaju naredbe u BASIC-jeziku. Za izgradnju ovih funkcija treba imati u vidu da je ograničavač azbučnog podatka u memoriji nula-bajt. To je bajt u kojem su svih 8 bita nule. Tako, ako se izvrši naredba

X$="BEOGRAD"

tada će polje u memoriji u kojem se registruje tekuća vrednost promenljive X$ imati sledeći izgled

B	E	O	G	R	A	D	•	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

gde je grafičkim simbolom • označen nula-bajt. Adresa polja u memoriji je programski dostupna pomoću funkcije PTR X$. Ako se ovo ima u vidu očigledno da nije teško izgraditi, raspoloživim konstrukcijama BASIC-jezika, razne funkcije nad azbučnim veličinama. Tako, sledeći potprogram vrši poređenje dva azbučna podatka koji su susedni elementi azbučnog niza:

```
3000 FOR K=0 TO 15
3010 A=BYTE(PTRX$(I)+K)
3020 B=BYTE(PTRX$(I+1)+K)
3030 IF A+B ELSE K=0 RETURN
```

```
3040 IF A<B K=—1:RETURN
3050 IF A>B K=1:RETURN
3060 NEXT K K=0:RETURN
```

rezultat poređenja se postavlja kao vrednost promenljive K, na sledeći način:

$$K=\begin{cases}-1 \text{ ako je } X\$(I)<X\$(I+1)\\ \ \ 0 \text{ ako je } X\$(I)=X\$(I+1)\\ +1 \text{ ako je } X\$(I)>X\$(I+1)\end{cases}$$

Ovaj potprogram se može uspešno koristiti za uređenje azbučnih nizova podataka u azbučni redosled. Ovo ćemo ilustrovati sledećim primerom.

Primer

Sastaviti program koji koristi gore navedeni potprogram za uređenje azbučnog niza podataka u azbučni redosled.

Azbučni podaci koje treba urediti neka se nalaze u programskoj datoteci. Biće to 6 prezimena koje treba izdati u azbučnom redosledu. Program se može zapisati u sledećem obliku:

```
10 !UREĐENJE AZBUČNOG NIZA
15 ARR$(5)
20 FOR I=0 TO 5
30 TAKE X$(I)
40 NEXT I
50 J=0
60 FOR I=0 TO 4
70 CALL 3000
80 IF K=1 X$=X$(I):X$(I)=X$(I+1):X$(I+1)=X$:J=1
90 NEXT I
100 IF J=1 GOTO 50
110 FOR I=0 TO 5
120 PRINT X$(I)
130 NEXT I
140 STOP
150 #"ŠOŠKIĆ","POPOVIĆ","POPOV"
160 #"ARSIĆ","MILIĆ","MILIČEVIĆ"
```

U prvom ciklusu (naredbe 20—40) izvrši se prenošenje azbučnih podataka iz programske datoteke u azbučni niz. Uređenje niza je organizovano razmenom elemenata niza ako ne zadovoljavaju azbučni redosled (naredba 80) sve dok konačno svi elementi ne budu u azbučnom redosledu. Kakav je odnos elemenata niza X$(I) i X$(I+1) utvrđuje se potprogramom koji se poziva naredbom u redu broj 70. Izdavanje uređenih elemenata niza vrši se u ciklusu od naredbe 110 do 130. Izvršavanjem ovog programa dobija se sledeći izveštaj:

```
> RUN
ARSIĆ
MILIĆ
MILIĆEVIĆ
POPOV
POPOVIĆ
ŠOŠKIĆ
>_
```

Kao što se vidi prezimena su poređena u azbučni redosled. Kod ovog uređenja treba imati u vidu da su slova Č, Ć, Ž i Š poslednja slova azbuke. Ovo sledi iz tabele 5.7.1 u kojoj su prikazane dekadne vrednosti pojedinih znakova azbuke u ASCII-kôdu.

12.3. Ušteda memorijskog prostora

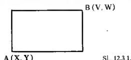
Računar GALAKSIJA raspolaže relativno malim kapacitetom RAM-memorije koja je na raspolaganju korisniku. Zato će često korisniku biti važno da zapiše program koji angažuje što je moguće manji prostor u memoriji računara. Memorijski prostor se može uštedeti ako se pri pisanju programa pridržavamo sledećih preporuka:

- u programu ne koristiti naredbe za komentare(!),
- sve službene reči, duže od dva slova, pisati samo sa početnim slovom iza kojeg sledi tačka,
- pisati veći broj naredbi u programskom redu,
- ne koristiti praznine između elementarnih konstrukcija u naredbi,
- koristiti potprograme kad god je to moguće.

Prve četiri preporuke, kada se primene, znatno smanjuju čitljivost programa, ali doprinose uštedi memorijskog prostora. Videli smo da broj reda može biti od jednocifrenog do petocifrenog broja. Ovo bi moglo da navede na pomisao da treba koristiti što je moguće manje brojeve redova. Međutim, ovo nije tačno iz razloga što se broj reda uvek registruje kao neoznačen binarni broj u jednom memorijskom registru (zauzima dva bajta).

Primer

Sastaviti program koji na ekranu crta pravougaonik za zadate koordinate tačaka A i B (sl. 12.3.1). Program ćemo sastaviti tako da se na početku programa unose koordinate tačaka A i B, zatim izvrši brisanje ekrana, a onda crtanje na ekranu. Crtanje će biti izvršeno tako da se najpre



Sl. 12.3.1.

crtaju horizontalne, a zatim vertikalne stranice pravougaonika. Na kraju se program završava dinamičkim zaustavljanjem računara. Prikazaćemo dva rešenja: u prvom će biti zapisan program sa svim potrebnim komentarima, tako da će program biti vrlo čitljiv, a u drugom zapisu biće ilustrovana mogućnost konciznog zapisa programa sa najmanjim zauzećem memorijskog prostora. Prvi zapis programa može biti:

```
10 !PROGRAM GI2.3.1
20 !PROGRAM CRTA PRAVOUGAONIK NA EKRANU
30 PRINT"UNESITE KOORDINATE TAČKE A "
40 INPUT X
50 INPUT Y
60 PRINT"UNESITE KOORDINATE TAČKE B "
70 INPUT V
80 INPUT W
90 !BRISANJE EKRANA
100 HOME
110 !CRTANJE HORIZONTALNIH STRANICA
120 FOR I=X TO V
130 DOT I,W
140 DOT I,Y
150 NEXT I
160 !CRTANJE VERTIKALNIH STRANICA
170 FOR J=W TO Y
180 DOT X,J
190 DOT V,J
200 NEXT J
210 GOTO 210
```

Koncizan zapis istog programa:

```
40I.X:I.Y:I.V:I.W:H.
120F.I=XTOV:D.I,W:D.I,Y:N.I
170F.J=WTOY:D.X,J:D.V,J:N.J
210G.210
```

Razlika u dužini programa je očigledna. Međutim, isto tako je vrlo očigledno da je koncizan zapis programa nečitljiv, teži za praćenje, testiranje i ispravljanje grešaka. Zato ovakav zapis ima smisla primeniti samo kada se radi o programima za koje inače ne postoji dovoljno prostora u memoriji, a takođe i za programe i potprograme koji se nalaze u biblioteci.

12.4. Razvijanje programa

Programiranje nije samo pisanje programa, kao što se to često misli. To je posao na izradi programa koji se sastoji od više faza.

1. Definisanje problema
2. Projektovanje programa
3. Pisanje programa

4. Ispitivanje programa
5. Izrada dokumentacije
6. Održavanje programa

Kao što se vidi pisanje programa je jedna od ovih 6 značajnih faza. Pri tome treba naglasiti da se pisanje programa može relativno brzo i sistematično naučiti, a za ostale faze to nije slučaj. Ovde ćemo ukratko opisati svaku od navedenih faza rada:

1. *Definisanje problema* obuhvata dobru postavku zadatka, definisanje ulaza, izlaza, obrade, mogućih izveštaja o greškama, kao i uzimanje u obzir ljudskih faktora pri eksploataciji programa.

2. *Projektovanje programa* obuhvata razradu definicije problema u oblik pogodan za pisanje programa. U ovoj fazi se koristi prikazivanje postupka rešavanja zadatka u obliku blok-šema, razbijanje zadatka na podzadatke u cilju primene modularnog programiranja, organizacija modula u sistem, kao i strukturno rešenje pojedinih modula.

3. *Pisanje programa* je faza u kojoj se projektovan program zapisuje u nekom od jezika programiranja. Pri ovome pored programskih rešenja koja obezbeđuju dobru strukturu i čitljivost programa treba imati u vidu i buduće ispitivanje i eksploataciju programa.

4. *Ispitivanje programa* ima za cilj proveru korektnosti programa, otkrivanje i ispravku grešaka u programu, kao i proveru test primera na računaru.

5. *Izrada dokumentacije* je važna faza rada, naročito za programe koji se daju na korišćenje drugim ljudima, kao što su veći aplikacioni i sistemski programi.

6. *Održavanje programa* je tesno vezano za dobru dokumentaciju programa. To je faza u kojoj se mogu unositi razna poboljšanja u programu ili proširenja programa. Sve izmene je neophodno prosledıti u dokumentaciji o programu.

Sve ove tačke u izradi programa su prisutne bez obzira koji jezik programiranja se koristi pri pisanju programa. Mi se nećemo opširnije baviti svim ovim fazama izrade programa, jer se pretpostavlja da su one jasne čitaocu. Međutim, ipak ćemo neke probleme istaći kada se radi o pisanju programa na BASIC-jeziku.

12.4.1. Pisanje programa

U fazi projektovanja programa neophodno je dobro sagledati programsko ostvarenje ideje rešavanja zadatka na računaru. Ovde se misli na izbor metode za obradu podataka, unošenje ulaznih podataka i izdavanje rezultata. Zatim uočavanje pojedinih podzadataka u okviru složenog zadatka. Podzadatke organizovati što više u obliku potprograma. Na ovaj način izvršiti što bolju modularizaciju programa. Kada je ova faza dobro razrađena može se pristupiti pisanju programa.

U pisanju programa važno je da svaki programski podzadatak bude ostvaren sa što boljom programskom strukturom. Zato treba, što je moguće više, poštovati principe strukturnog programiranja, a to znači da se svaki program gradi kao kompozicija tri elementarne strukture: linijske, razgranate i cikličke, pri čemu svaka od elementarnih struktura ima jedan ulaz i jedan izlaz. Međutim, pored strukture programa ima još dosta elemenata koji utiču na čitljivost i razumljivost programa. To su zapisi strukture podataka koje se organizuju u programu, sadržaj komentara i dr. Dakle, program sa istom funkcijom, može biti zapisan na mnogo različitih načina. Način na koji će programer izraziti ono što treba da radi jedan program čini *programski stil*. Dobar programski stil je onaj koji obezbeđuje dobru strukturu, čitljivost i razumljivost programa. Mi ćemo se u sledećem izlaganju upoznati sa elementima koji mogu doprineti dobrom programskom stilu pri pisanju programa na BASIC-jeziku.

Modularnost programa

Pri pisanju velikih programskih sistema, jedna od bitnih komponenata u izgradnji sistema jeste modularnost programa. Modul predstavlja podzadatak koji ima svoju funkcionalnu celinu u celom sistemu. Istina je da treba dobro razmisliti pri razbijanju zadatka na podzadatke, ali to je zapravo jedini put da se uspešno ostvari veliki programski sistem, kako se stanovišta izgradnje i testiranja, tako i sa stanovišta održavanja i eksploatacije. Programski modul treba da sadrži 20 do 50 programskih redova. Naravno, ne treba se truditi da se mali programski zadaci razbiju na module. Modularnost se najbolje sprovodi kroz koncept potprograma, tako da potprogram čini modul.

Struktuiranost modula

Dobra struktuiranost svakog modula znatno će doprineti lakšem testiranju i razumevanju funkcije modula. Ovde se, pre svega, misli na kompoziciju elementarnih struktura u programskom ostvarenju modula, prema načelima strukturnog programiranja. Međutim, ako programer želi da ostvari izuzetno duhovita rešenja, koja nisu preporučljiva sa gledišta strukturnog programiranja, tada ih treba posebnim komentarima dobro objasniti. U programima treba, po pravilu izbegavati programsku modifikaciju, jer ova dovodi do teško razumljivih programa, a posebno otežava testiranje programa. Pri ostvarenju programskih modula voditi računa da se programski kontrolišu razne neregularne situacije koje mogu nastati i o ovome davati korisniku jasne izveštaje. Ovde se misli na regularnost ulaznih podataka, zatim nedozvoljene operacije, kao što je izračunavanje kvadratnog korena iz negativnog broja, deljenje nulom, pojava prekoračenja i sl.

Struktuiranost podataka

U programiranju se dosta govori o struktuiranosti programa, a manje o dobroj struktuiranosti podataka. Međutim, dobra struktuiranost podataka može znatno da doprinese izgradnji efikasnijih i jasnijih programa. Navodimo neke elemente koji opravdavaju ovu tvrdnju:

- organizovati podatke u okviru niza tako da se prolazak kroz podatke može organizovati u okviru ciklusa,
- podatke treba združiti u jednu celinu u zapisu programske datoteke, a ne razbacivati ih po celom programu,
- kod programiranja na mašinskom jeziku, organizovati podatke tako da se što bolje mogu iskoristiti raspoloživi načini adresiranja u mašinskom jeziku.

Standardizacija forme

Standardizacija forme zapisa programa, takođe, doprinosi boljoj preglednosti programa. Ovo je naročito korisno kod pisanja dokumentacije o programu. U ovom smislu bismo mogli navesti sledeće:

- na početku programa, u obliku komentara, opisati funkciju programa,
- ako se koristi azbučni niz navesti naredbu dimenzionisanja na početku programa pre drugih izvršnih naredbi,
- navesti varijante o mogućem izvršavanju programa (meni),
- program završiti naredbom STOP ili dinamičnim zaustavljanjem računara. Na ovaj način će biti jasnije da li je ceo program prisutan u memoriji računara.

Komentari

Programeri se pri pisanju programa usredsrede na pisanje naredbi, a zaboravljaju značaj komentara u programu. Kasnije obično nemaju ni volje ni vremena da dopišu komentare. Međutim, komentari su korisni u fazi testiranja programa, a neophodni u dokumentaciji programa. O pisanju komentara može se reći sledeće:

- svaka programska jedinica treba da ima uvodni komentar u kojem se navodi funkcija programske jedinice i značajni parametri,
- za veće programske sisteme na početku treba navesti ime autora programa, datum izrade i datum poslednjih izmena u programu,
- komentari koji se pišu u programu treba da objašnjavaju ostvarenje postupka rešavanja zadatka, a ne da tumače naredbe.

12.4.2. Ispitivanje programa

Kada je program napisan predstoji faza ispitivanja korektnosti programa. Ovo je važan, a često i težak posao

u programiranju Dakle, problem je dokazati da napisan program rešava zadatak koji smo želeli da rešimo. Bilo bi lepo da možemo teorijski dokazati da program rešava postavljen zadatak. Ovakva teorijska disciplina se razvija i postoje izvesni rezultati na ovom planu [6]. Međutim, ovi rezultati nisu takvi da imaju veći značaj u praktičnom programiranju. Ostaje problem da se programer uveri u ispravnost svog programa. Ovo se danas radi tako što programer ispituje program na karakterističnim primerima. Ako se program ponaša korektno za izabrane primere, smatramo da je program ispravan. Međutim, jasno je da na ovaj način ne možemo dokazati korektnost, već samo nekorektnost programa. Dakle, u ovom odeljku želimo da ukažemo na neke elemente koji mogu doprineti lakšem i bržem ispitivanju programa. Ispitivanje programa sadrži sledeće faze:

- otkrivanje grešaka u programu,
- otklanjanje otkrivenih grešaka i
- merenje performansi programa.

Prve dve faze mogli bi nazvati *češljanje programa* (eng. debugging), a treću fazu *testiranje programa* (eng. testing). Odmah da kažemo da se često za sve tri faze kaže testiranje programa. Međutim, čitalac će se lako složiti da se treća faza bitno razlikuje od prve dve. Dakle, kroz prve dve faze mi želimo da dobijemo korektan program, a u trećoj fazi želimo, kroz test primere ih merenjem, da utvrdimo neke važne eksploatacione karakteristike programa. Naravno, ove karakteristike ima smisla utvrđivati samo za korektan program. U mnogim primenama, trećoj fazi se ne poklanja veća pažnja. Međutim, lako ćemo zaključiti da ima elemenata koje treba oceniti u ovoj trećoj fazi. Recimo, vreme trajanja programa, vreme reakcije programa na poruke korisnika, valjanost izveštaja za interaktivni rad, angažovanje memorijskog prostora, mogućnosti programa za rad u realnom vremenu (posebno u kontroli procesa) i sl. U ovom izlaganju mi ćemo se baviti, pre svega, problemom otkrivanja i otklanjanja grešaka u programu. Aktivnosti koje programer može sprovesti u ovom smislu možemo podeliti na četiri grupe:

- priprema za ispitivanje programa,
- programska podrška ispitivanju programa,
- otkrivanje grešaka i
- otklanjanje grešaka.

Ove grupe ne treba shvatiti kao odvojene poslove, već se u fazi ispitivanja programa aktivnosti iz ove četiri grupe međusobno prepliću.

Priprema za ispitivanje programa

Programer mora izvršiti solidnu pripremu za ispitivanje programa, pre nego što je došao na računar da ispituje program. Nema smisla doći na računar, radi ispitivanja programa, bez ikakve ideje kako pristupiti ispitivanju. Zato programer treba da ima u vidu sledeće:

- u fazi pisanja programa voditi računa da predstoji ispitivanje program. Sprovođenjem koncepcija modular-

nosti i dobre strukturanosti olakšaće se ispitivanje programa,

● pripremiti ulazne podatke sa kojima će se ispitivati program i odgovarajuće rezultate, tako da se može lako i brzo zaključiti da li se program izvršava korektno,

● sprovesti ručno ispitivanje programa za neke specijalne slučajeve, kao što su: da li se ciklus ne izvršava, da li se korektno jedanputa izvršava da li je korektno grananje i sl

Programska podrška ispitivanju programa

Za potrebe ispitivanja programa često se u programskom sistemu nalaze posebni programi koji omogućuju lako ispitivanje programa Ovo naročito važi za simbolički nivo programiranja. Na računaru GALAKSIJA u ovom smislu ćemo ukazati na:

1 Komandu EDIT koja omogućuje lako ispravljanje grešaka unutar jednog programskog reda.

2 Komanda RUN n koja omogućuje izvršavanje programa od zadatog broja reda (n)

3 Snimanje celog programa ili delova programa na kaseti (komanda SAVE) Korisnik kad god razvija program treba da ima jednu radnu verziju programa na kaseti Tako, ako ne može da završi rad na razvoju programa jednog dana, može unošenje i testiranje programa da nastavi sledećeg dana.

Otkrivanje grešaka

Ovo je najvažnija faza ispitivanja programa Naravno ne može se dati detaljno uputstvo kako otkriti grešku u programu, jer je broj i raznovrsnost mogućih grešaka ogroman Međutim, mogu se dati neka opšta uputstva koja mogu biti od koristi:

● u programu obavezno vršiti inicijalizaciju promenljivih ne oslanjajući se na neke pretpostavljene vrednosti, npr smatrati da su pre izvršenja programa vrednosti promenljivih nule,

● proverite da li se dobro postavljaju i ažuriraju brojači u programu, kao i drugi pokazivači, na primer pokazivač programske datoteke.

● lista mogućih grešaka u programu je neograničena Uprkos tome u praksi se programeri brzo obuče da lociraju i otkrivaju greške u programu bez obzira na njihovu raznovrsnost

Otklanjanje grešaka

Svakako bi bilo najbolje otkriti sve greške u programu, a zatim ih ispraviti Međutim, najčešće programer ne može ovako uspešno da obavi ispravke u programu Po otkrivanju nekih grešaka vrši se njihovo otklanjanje, a zatim se

ponovo traga za drugim greškama itd. Kod otklanjanja grešaka treba imati u vidu sledeće:

- otkrivenu grešku treba u potpunosti razumeti, u čemu se sastoji i kakve efekte ima na rad programa. Ovakvu grešku treba otkloniti. Međutim, ne treba na brzinu doneti zaključak o grešci i otklanjati je, jer preti opasnost da se ispravni delovi programa učine pogrešnim,
- svaku otkrivenu grešku odmah ispraviti,
- ne zaboraviti da pri ispravljanju programa treba ispraviti i odgovarajuće komentare.

Na kraju, kada su sve greške otklonjene, treba za potrebe dokumentacije uraditi nekoliko karakterističnih test primera kojima se ilustruje rad i način korišćenja programa.

12.4.3. Izrada dokumentacije

Većina programera ne voli pisanje dokumentacije o programu, jer smatra da je rad na programu završen onda kada program ispravno radi na računaru. Međutim, dokumentacija je neophodna kako korisnicima programa, tako i autoru programa. Ako je potrebno vršiti bilo kakve izmene ili dopune programa, to se bez dobre dokumentacije ne može sprovesti, čak ni od strane autora programa. Dokumentacija treba da sadrži:

- opis funkcije programa,
- teorijske osnove na kojima počiva program, sa spiskom literature,
- opštu blok-šemu programa u kojoj se vide veze između pojedinih modula programa,
- spisak potprograma sa opisom njihovih funkcija, posebno spisak zahtevanih potprograma iz biblioteke,
- opis ulaznih podataka i izlaznih rezultata, kao i opis važnijih struktura podataka u programu,
- program sa test primerima,
- opis performansi programa,
- uputstvo za korišćenje programa i
- opis mogućih izmena i dopuna programa.

Naravno, izgled dokumentacije u velikoj meri zavisi od namene i veličine programa. Programi koji čine veće programske sisteme mogu imati odvojeno pisanu dokumentaciju za instaliranje programa na računarskom sistemu, održavanje i korišćenje programa.

12.5 Biblioteke potprograma

Biblioteke potprograma predstavljaju veliku pomoć pri izradi programa. Koristeći biblioteku korisnik brže dolazi do programa, jer ne unosi preko tastature potprograme koji su već u biblioteci i brže testira program jer su potprogrami već provereni. Biblioteka potprograma može biti razvijena od strane korisnika ili proizvođača računara. Prema tome na kom jeziku su napisani potprogrami biblioteka može biti: biblioteka BASIC-potprograma ili biblioteka mašinskih potprograma.

Biblioteka BASIC-potprograma

To su potprogrami koji se pozivaju naredbom CALL sa brojem prvog programskog reda u potprogramu. Ovakve potprograme treba programirati sa velikim brojevima redova (na primer preko 5000). Kada je sastavljena biblioteka tada se na kaseti snime samo potprogrami biblioteke. Kada se želi razviti program koji koristi potprograme biblioteke tada se prvo biblioteka prenese sa kasete u zonu programa, a zatim unese program koji se razvija, izvrši njegovo testiranje i snimi program sa bibliotekom na kasetu (SAVE). Ovo nešto poveća dužinu programa na traci, jer je pridružena i biblioteka, ali predstavlja jednostavan način da korisnik upotrebi biblioteku BASIC-potprograma. Pri pisanju potprograma za biblioteku, treba poštovati sledeće preporuke:

- birati velike brojeve programskih redova, tako da ne dolazi do preklapanja ovih brojeva sa brojevima redova u programu,
- u potprogramu, po mogućstvu, koristiti samo promenljive koje su ulazne, odnosno izlazne veličine potprograma,
- standardizovati imena promenljivih koje se koriste u potprogramima biblioteke,
- ne programirati ulaz i izlaz u potprogramima, osim ako to nisu potprogrami čija je ovo jedina funkcija,
- obezbediti što manje angažovanje memorijskog prostora za potrebe biblioteke.

Poslednja preporuka se može ostvariti ako se ne koriste komentari u potprogramima, praznine unutar programskog reda i ako se koristi skraćivanje službenih reči (pisanjem samo početnog slova sa tačkom).

Biblioteka mašinskih potprograma

U ovom slučaju korisnik može postupiti na sledeći način: Rezerviše memorijski prostor za mašinske potprograme primenjujući komandu NEW n, a zatim se unošenje potprograma može izvršiti izvršavanjem naredbi BYTE, a još bolje pomoću BASIC-programa u kojem se kao programska datoteka nalazi potprogram na mašinskom jeziku. Ovo je primenjeno u zadatku 11.7.1. Kada je unošenje potprograma i njihovo testiranje završeno uništi se BASIC-program naredba po naredba (unošenjem samo brojeva redova) ili naredbom WORD &2C38, WORD(&2C36) ili komandom NEW n, gde je n isti dekadni broj sa kojim je prethodno izvršeno pomeranje početka BASIC-programa. Posle ovoga biblioteka mašinskih potprograma može biti snimljena komandom SAVE. Kada se želi razviti BASIC-program

koji koristi ovakvu biblioteku, tada se prvo unese biblioteka sa kasete u memoriju komandom OLD, a zatim korisnik može unositi i testirati BASIC-program. Ovde je važno primetiti da korisnik ne sme da primeni komandu NEW za vreme razvoja programa, jer bi izbrisao i biblioteku mašinskih potprograma. Kada je program razvijen može se snimiti zajedno sa bibliotekom na kaseti, a zatim po potrebi koristiti.

OPIS PROGRAMSKIH JEZIKA

Iako su veštački jezici znatno siromašniji od prirodnih jezika, oni se grade na sličan način. Najpre se uvodi azbuka jezika. Ređanjem simbola azbuke grade se konstrukcije u jeziku. Proizvoljan niz simbola iz azbuke jezika zvaćemo *niska*. Važno je uočiti da neke niske pripadaju jeziku, a neke ne pripadaju jeziku. Tako je srpskohrvatski jezik izgrađen nad azbukom od 30 slova. Neke niske slova predstavljaju reči u jeziku, a neke niske ne čine reči. Na sličan način neki nizovi reči čine rečenice, a neki ne čine rečenice. Nauka o jeziku koja izučava koje niske simbola čine pravilne konstrukcije u jeziku zove se *sintaksa*. Sintaksa prirodnih jezika definisana je gramatikom jezika. Gramatiku jezika čini skup pravila, na osnovu kojih se za svaku nisku simbola u jeziku može reći da li niska pripada ili ne pripada jeziku. Međutim, konstrukcije u jeziku imaju i određeno značenje. Nauka o jeziku koja izučava značenje konstrukcija u jeziku zove se *semantika*. Mnoge ideje i pojmovi iz prirodnih jezika prenose se i koriste u veštačkim jezicima, a takođe razvoj teorije formalnih jezika doprinosi izučavanju prirodnih jezika.

Jedan od važnih problema u izučavanju programskih jezika jeste opis sintakse i semantike jezika. Programski jezici služe za zapis algoritma po kojem računar treba da izvršava niz operacija pri rešavanju određenog zadatka. Prema tome, to su jezici u kojima sve mora biti zapisano precizno i bez ikakvih dvosmislenosti. Jednoznačno definisanje sintakse i sementike programskih jezika značajno je kako za korisnike, tako i za konstruktore programskih jezika. Danas su posebno dobro razvijeni jezici za opis sintakse programskih jezika. Za jezik koji služi za opis sintakse programskog jezika kažemo da je *metaprogramski jezik*. U ovoj knjizi koristi se tzv. Bekusova notacija kao metaprogramski jezik, za opis programskog jezika BASIC.

Neka su J i M dva jezika. Ako se jezik J opisuje jezikom M onda je J jezik, a M metajezik. Niske simbola jezika J zovu se *metakonstante* u jeziku M. Pojmovi koji se definišu u jeziku J zovu se metapromenljive u jeziku M. Metapromenljiva se piše kao tekst između ugaonih zagrada. Metakonstanta se može dodeliti kao vrednost metapromenljivoj. *Metaizraz* je sastavljen od metakonstanti i metapromenljivih među sobom razdvojenih metaoperacijama. Metaoperacija može biti operacija spajanja ili razdvajanja. Operacija *spajanja* piše se u obliku

$$\alpha\beta$$

a čita se "α za kojim sledi β", gde su α i β metakonstante, metapromenljive ili metaizrazi. Operacija razdvajanja piše se u obliku

$$\alpha \mid \beta$$

a čita se "α ili β", gde su α i β metakonstante, metapromenljive ili metaizrazi. *Metaformula ili definiciona jednačina* sadrži metapromenljivu i metaizraz među sobom razdvojene simbolom za operaciju *definisanja* (::=), tj.

$$\gamma ::= \psi$$

i čita se "γ to je ψ" gde je γ metapromenljiva a ψ metaizraz. Metapromenljiva γ, na levoj strani definicione jednačine, predstavlja pojam koji se definiše u jeziku J. Metaizraz ψ generiše metakonstatntu, odnosno nisku simbola u jeziku J, koja može biti dodeljena metapromenljivoj γ. Broj simbola u niski određuje *dužinu niske*. Niska koja ne sadrži nijedan simbol zove se *prazna niska* i ima dužinu 0. Da bismo omogućili kraći zapis definicija u jeziku uvešćemo sledece oznake.

$$[\alpha]_i^j = \underbrace{\alpha\alpha\ldots\alpha}_{i\text{—puta}} \mid \underbrace{\alpha\alpha\ldots\alpha}_{i+1\text{—puta}} \mid \ldots \mid \underbrace{\alpha\alpha\ldots\alpha}_{j\text{—puta}}$$

gde je $i>0$, $j \geqslant i$. Za $i=0$ važi:

$$[\alpha]_0^j = e \mid \alpha \mid \alpha\alpha \mid \ldots \mid \underbrace{\alpha\alpha\ldots\alpha}_{j\text{—puta}}$$

gde je e prazna niska, a α element jezika. Takođe, važe oznake:

$$[\alpha]_0^\infty = [\alpha]^*$$
$$[\alpha]_1^\infty = [\alpha]^+$$

pri čemu je u svakoj implementaciji (ostvarenju na računaru) programskog jezika gornja granica ponavljanja ograničena. Tako ce ovo ograničenje u implementaciji BASIC-jezika biti dužina jednog reda (125 znakova). Kako vrlo često u konstrukcijama programskih jezika neki element jezika može biti naveden ili izostavljen koristi se i oznaka

$$[\alpha]_0^1 = [\alpha]$$

Da bismo povećali čitljivost definicija uvodimo i konvenciju što znači da se bira jedan od navedenih elemenata

$$\left\{\begin{matrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{matrix}\right\} = \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$$

Bekusova notacija se koristi za definisanje sintakse programskog jezika. Za opis semantike koristi se opis akcije koju proizvodi određena konstrukcija jezika na računaru. Akcija se opisuje na srpskohrvatskom jeziku.

Primer 1

Primenu Bekusove notacije objasnićemo na uprošćenom primeru izgradnje prostih rečenica u prirodnom jeziku. Znači, pojam koji želimo da definišemo jeste prosta rečenica i to će biti metapromenljiva u Bekusovoj notaciji. Da bismo lakše definisali pravila izgradnje složenih konstrukcija u prirodnom jeziku uvodimo tzv. *gramatičke kategorije*, kao što su subjekat, predikat, imenica, zamenica, glagol itd. Sve gramatičke kategorije javljaju se kao metapromenljive u Bekusovoj notaciji.

Tako se uprošćena sintaksa prostih rečenica može zapisati u obliku

⟨prosta rečenica⟩ ::= ⟨subjekat⟩□⟨predikat⟩.
⟨subjekat⟩ ::= ⟨imenica⟩|⟨zamenica⟩
⟨imenica⟩ ::= Mirko | Darko | Milica | Jovan | Pera | Danica | Radomir
⟨zamenica⟩ ::= On | Ona | Ono
⟨predikat⟩ ::= ⟨glagol⟩
⟨glagol⟩ ::= trči | radi | stoji | uči | čita | piše | peva | skače

Ovde smo uzeli da je jezik J skup prostih rečenica. Azbuka jezika J je skup velikih i malih slova, praznina (označen sa ␣) i tačka. Navedena gramatika generiše neke proste rečenice, kao

Mirko trči
Ona piše
Radomir peva

Naravno, teško bi bilo opisati sintaksu prirodnog jezika u opisanoj notaciji, ali ovaj primer ilustruje smisao uvedene notacije.

Primer 2. Koristeći Bekusovu notaciju opisati zapis celih brojeva u dekadnom brojnom sistemu. Ceo broj možemo definisati na sledeći način:

⟨ceo broj⟩ ::= [±][⟨cifra⟩]$^+$
⟨cifra⟩ ::= 0|1|2|3|4|5|6|7|8|9

Navešćemo neke primere celih brojeva koje generiše navedena gramatika:

124   —5   +508   0   —0   —0140

Na sličan način se mogu opisati i svi drugi pojmovi u programskom jeziku, kao što su promenljive, izrazi, naredbe, potprogrami i programi.

KOMANDE INTERPRETATORA I TASTATURNE KOMANDE

KOMANDE INTERPRETATORA

Komanda za ispravljanje naredbi

EDIT ⟨broj reda⟩

Značenje: Briše ekran i programski rad se navedenim brojem reda izdaje na početku ekrana. Tasteri označeni strelicama nalevo i nadesno omogućuju pomeranje pokazivača pozicije na ekranu, a taster DEL brisanje znakova desno od pokazivača. Ubacivanje novog teksta vrši se od pozicije pokazivača jednostavnim kucanjem na tastaturi. Ispravljanje se završava pritiskom na taster ENTER.

Komanda za izdavanje programa

LIST [⟨broj reda⟩]

Značenje: Ako nije naveden broj reda izdaje BASIC-program od početka programa do kraja programa na ekranu. Ako je naveden broj reda tada se izdaje program od zadatog broja reda do kraja programa. Izdavanje se vrši samo za vreme dok je taster ENTER ili STOP/LIST pritisnut.

Komanda za brisanje programa

NEW [⟨dekadni broj⟩]

Značenje: Briše BASIC-program u zoni programa i postavlja početak zone programa od, u sistemu, pretpostavljene adrese. Ako je naveden dekadni broj tada se početak zone programa pomera za navedeni broj bajtova prema višim adresama u RAM-memoriji.

Komanda za učitavanje sa trake

OLD [⟨dekadni broj⟩]

Značenje: Unosi sadržaj sa kasete u RAM-memoriju. Ako je naveden dekadni broj tada se sadržaj sa trake učitava u zonu memorije čija je početna adresa pomerena u odnosu na pretpostavljenu vrednost početne adrese zone programa, za broj bajtova ukazan dekadnim brojem u komandi OLD.

Komanda za verifikaciju snimljenog programa

OLD?

Značenje: Vrši čitanje programa sa trake i poredi ga sa programom u memoriji. Ako ne postoji razlika verifikacija se završava bez poruke. Ako postoji razlika između programa na traci i programa u memoriji tada se javlja poruka WHAT?

Komanda za izvršavanje programa

RUN [(broj reda)]

Značenje: Vrši prelazak na izvršavanje programa od početka programa. Ako je naveden broj reda tada izvršavanje programa počinje od navedenog broja reda.

Komanda za izvršavanje programa

SAVE [(brojni podatak),(brojni podatak)]

Značenje: Vrši snimanje programa iz zone programa, od pretpostavljene početne adrese zone programa (&2C3A), na traku. Ako su navedeni brojni podaci, onda ovi definišu početnu i krajnju adresu zone u memoriji, čiji se sadržaj snima.

TASTATURNE KOMANDE

Brisanje ekrana i unetog znaka

SHIFT/DEL — briše ekran i postavlja pokazivač pozicije na ekranu na početak prvog reda na ekranu,

→ — briše poslednji uneti znak u tekućem ulaznom slogu.

Unošenje naredbi i komandi

ENTER	—	unosi tekući ulazni slog i postavlja pokazivač pozicije na ekranu na početak sledećeg reda.

Izdavanje programa

LIST	—	izdaje program iz zone programa na ekranu za vreme dok se taster drži pritisnut

Ispravke naredbi komandom EDIT

←	—	pomera pokazivač pozicije za jedno mesto ulevo na ekranu,
→	—	pomera pokazivač pozicije za jedno mesto udesno na ekranu,
DEL	—	briše znak desno od pokazivača pozicije na ekranu,
ENTER	—	završava ispravljanje programskog reda i ispravljen programski red unosi u zonu programa.

Višestruko unošenje

REPT	—	ponavlja poslednji uneti znak za vreme dok je taster pritisnut.

Komande od značaja za vreme izvršavanja programa

DEL	—	suspenduju izvršavanje programa za vreme dok je taster pritisnut.
BRK	—	prekida izvršavanje programa i izdaje izveštaj BREAK (broj reda) gde je broj reda, broj programskog reda koji treba da se izvrši kao sledeći programski red u programu.

SINTAKSA I SEMANTIKA BASIC-JEZIKA

AZBUKA

⟨simbol⟩: :=⟨osnovni simbol⟩ | ⟨izveden simbol⟩
⟨osnovni simbol⟩: :=⟨slovo⟩ | ⟨cifra⟩ | ⟨specijalni znak⟩ | ⟨YU-slovo⟩

⟨slovo⟩: := A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
⟨YU-slovo⟩: := Ć | Č | Ž | Š
⟨cifra⟩: :=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

⟨specijalni znak⟩: := { ⟨interpunkcijski znak⟩ ⟨aritmetički znak⟩ ⟨relacijski znak⟩ ⟨pomoćni znak⟩ ⟨grafički simbol⟩ }

⟨interpunkcijski znak⟩: := ! | " | (|) | — | : ; . ?

⟨aritmetički znak⟩: := + | — | * | /
⟨relacijski znak⟩: := < | > | =
⟨pomoćni znak⟩: := # | $ | % | &
⟨grafički simbol⟩: :=█

⟨izveden simbol⟩: :=⟨izveden relacijski simbol⟩ | ⟨službena reč⟩
⟨izveden relacijski simbol⟩: := EO
⟨službena reč⟩: : = ARR$ | BYTE | CALL | DOT | ELSE | FOR | GOTO | HOME | IF | INPUT | NEXT | PRINT RETURN | STEP | STOP | TAKE | TO | UNDOT | WORD

Primedba: YU-slova se proizvode na tastaturi na sledeći način:

SHIFT/C proizvodi Ć
SHIFT/X proizvodi Č
SHIFT/Z proizvodi Ž
SHIFT/S proizvodi Š

Svaka službena reč duža od dva slova, pri pisanju programa, može se skratiti tako da se piše samo prvo slovo iza kojeg se navodi obavezno tačka

$$\langle\text{elementarna sintaksna jedinica}\rangle ::= \left\{\begin{matrix}\langle\text{podatak}\rangle\\ \langle\text{promenljiva}\rangle\\ \langle\text{niz}\rangle\\ \langle\text{izraz}\rangle\end{matrix}\right\}$$

$$\langle\text{podatak}\rangle ::= \left\{\begin{matrix}\langle\text{brojni podatak}\rangle\\ \langle\text{azbučni podatak}\rangle\end{matrix}\right\}$$

$$\langle\text{brojni podatak}\rangle ::= \left\{\begin{matrix}\langle\text{dekadni broj}\rangle\\ \langle\text{heksadekadni broj}\rangle\end{matrix}\right\}$$

$$\langle\text{dekadni broj}\rangle ::= \left\{\begin{matrix}\langle\text{pozicioni zapis}\rangle\\ \langle\text{eksponencijalni zapis}\rangle\end{matrix}\right\}$$

$$\langle\text{pozicioni zapis}\rangle ::= \left\{[\pm]\ \begin{matrix}[\langle\text{cifra}\rangle]_1^* & [\ ]\\ [\langle\text{cifra}\rangle]_0^* & .[\langle\text{cifra}\rangle]_1^*\end{matrix}\right\}$$

$$\langle\text{eksponencijalni zapis}\rangle ::= \langle\text{pozicioni zapis}\ \text{E}\ [\pm][\langle\text{cifra}\rangle]_1^2$$

$$\langle\text{heksadekadni broj}\rangle ::= \&[\langle\text{heksadekadna cifra}\rangle]_1$$

$$\langle\text{heksadekadna cifra}\rangle ::= 0\ |\ 1\ |\ 2\ |\ 3\ |\ 4\ |\ 5\ |\ 6\ |\ 7\ |\ 8\ |\ 9\ |\ A\ |\ B\ |\ C\ |\ D\ |\ E\ |\ F$$

⟨azbučni podatak⟩ : := " [⟨osnovni simbol⟩]* "

$$\langle\text{promenljiva}\rangle ::= \left\{\begin{matrix}\langle\text{brojna promenljiva}\rangle\\ \langle\text{azbučna promenljiva}\rangle\end{matrix}\right\}$$

⟨brojna promenljiva⟩ : := ⟨slovo⟩

⟨azbučna promenljiva⟩ : := X$ | Y$

$$\langle\text{niz}\rangle ::= \left\{\begin{matrix}\langle\text{brojni niz}\rangle\\ \langle\text{azbučni niz}\rangle\end{matrix}\right\}$$

⟨brojni niz⟩ : := A

⟨azbučni niz⟩ : := X$

$$\langle\text{indeksna promenljiva}\rangle ::= \left\{\begin{matrix}\langle\text{indeksna brojna promenljiva}\rangle\\ \langle\text{indeksna azbučna promenljiva}\rangle\end{matrix}\right\}$$

⟨indeksna brojna promenljiva⟩ : := A (⟨indeks⟩)

⟨indeksna azbučna promenljiva⟩ : := X$ (⟨indeks⟩)

⟨indeks⟩ : := ⟨brojni izraz⟩

$$\langle\text{izraz}\rangle ::= \left\{\begin{matrix}\langle\text{brojni izraz}\rangle\\ \langle\text{azbučni izraz}\rangle\\ \langle\text{relacijski izraz}\rangle\end{matrix}\right\}$$

⟨brojni izraz⟩ : := [±]⟨član⟩ [{±}⟨član⟩]*

```
⟨član〉: :=⟨faktor⟩[{*/}⟨faktor⟩]*

                 ⎧ ⟨brojni podatak⟩
                 ⎪ ⟨brojna promenljiva⟩
⟨faktor⟩: :=     ⎨ ⟨indeksna brojna promenljiva⟩
                 ⎪ ⟨numerička funkcija⟩
                 ⎩ (⟨brojni izraz⟩)

                         ⎧ INT (⟨brojni izraz⟩)
                         ⎪ RND
                         ⎪ VAL (⟨brojni izraz⟩)
                         ⎪ PTR [⟨promenljiva⟩]
⟨numerička funkcija⟩: := ⎨ KEY (0)
                         ⎪ MEM
                         ⎪ BYTE (⟨brojni izraz⟩)
                         ⎪ WORD (⟨brojni izraz⟩)
                         ⎩ USR (⟨brojni izraz⟩)
```

Značenje numeričkih funkcija

1. Vrednost funkcije INT je najveći ceo broj jednak ili manji od vrednosti argumenta funkcije
2. Funkcija RND generiše pseudo-slučajni broj iz intervala (0, 1)
3. Funkcija VAL izračunava vrednost izraza, čija je adresa u memoriji argument funkcije, a vrednost funkcije izračunata vrednost izraza.
4. Vrednost funkcije PTR je adresa promenljive koja je navedena kao argument funkcije Ako argument nije naveden vrednost funkcije je adresa bajta u memoriji koji sledi iza imena funkcije (PTR).
5. Vrednost funkcije MEM je broj slobodnih bajtova u RAM-memoriji
6. Vrednost funkcije KEY(0) je ASCII-vrednost unetog znaka sa tastature
7. Vrednost funkcije BYTE je sadržaj bajta, adresiranog argumentom funkcije, uzet kao neoznačen broj
8. Vrednost funkcije WORD je sadržaj reči, adresirane argumentom funkcije, uzeta kao označen broj
9. Funkcija USR vrši prelazak na potprogram na mašinskom jeziku počev od adrese koja je određena celobrojnom vrednošcu brojnog izraza Vrednost funkcije je sadržaj registra HL mikroprocesoru Z80A

```
                  ⎧ ⟨azbučni podatak⟩          ⎫   ⎡  ⎧ ⟨azbučni podatak⟩          ⎫ ⎤*
                  ⎪ ⟨azbučna promenljiva⟩      ⎪   ⎢  ⎪ ⟨azbučna promenljiva⟩      ⎪ ⎥
⟨azbučni izraz⟩: :=⎨ ⟨indeksna azbučna         ⎬   ⎢+ ⎨ ⟨indeksna azbučna          ⎬ ⎥
                  ⎪ promenljiva⟩               ⎪   ⎢  ⎪ promenljiva⟩               ⎪ ⎥
                  ⎩ ⟨azbučna funkcija⟩         ⎭   ⎣  ⎩ ⟨azbučna funkcija⟩         ⎭ ⎦

⟨azbučna funkcija⟩: :=CHR$ (⟨brojni izraz⟩)
```

Značenje azbučne funkcije

Vrednost funkcije CHR$ je znak u ASCII-kodu koji odgovara celobrojnoj vrednosti argumenta

⟨relacijski izraz⟩: := { ⟨brojni relacijski izraz⟩ | ⟨azbučni relacijski izraz⟩ | ⟨tasturni relacijski izraz⟩ | ⟨grafički relacijski izraz⟩ }

⟨brojni relacijski izraz⟩: := ⟨brojni izraz⟩ $\left\{ \begin{matrix} \geq \\ \leq \\ = \end{matrix} \right\}$ ⟨brojni izraz⟩

⟨azbučni relacijski izraz⟩: := EQ { ⟨azbučna promenljiva⟩ | ⟨indeksna azbučna promenljiva⟩ } [,] { ⟨azbučna promenljiva⟩ | ⟨indeksna azbučna promenljiva⟩ }

⟨tastaturni relacijski izraz⟩: := KEY (⟨brojni izraz⟩)

⟨grafički relacijski izraz⟩: := DOT(⟨brojni izraz⟩, ⟨brojni izraz⟩)

Značenje relacijskih izraza:

1. Ako su brojni izrazi u navedenoj relaciji tada brojni relacijski izraz ima vrednost 1 u protivnom 0.
2. Ako su navedene azbučne veličine jednake tada azbučni relacijski izraz ima vrednost 1 u protivnom 0
3. Tastaturni relacijski izraz ima vrednost 1, ako je celobrojna vrednost izraza jednaka pridruženom broju (tabela 6.1.1) pritisnutog tastera u protivnom relacijski izraz ima vrednost 0
4. Grafički relacijski izraz ima vrednost 1 ako je na ekranu osvetljena tačka sa navedenim koordinatama, u suprotnom vrednost izraza je 0

SLOŽENE SINTAKSNE JEDINICE

⟨složena sintaksne jedinice⟩: := { ⟨program⟩ | ⟨potprogram⟩ | ⟨programski red⟩ | ⟨naredba⟩ }

⟨program⟩: := [⟨označen programski red⟩]$^{+}$

⟨potprogram⟩: := [⟨označen programski red⟩]$^{+}$ ⟨broj reda⟩ RETURN

⟨programski red⟩ : := { ⟨neoznačen programski red⟩ | ⟨označen programski red⟩ }

⟨neoznačen programski red⟩: : = ⟨naredba⟩ [:⟨naredba⟩]*

⟨označen programski red⟩: : = ⟨broj reda⟩⟨neoznačen programski red⟩

⟨broj reda⟩: : = [⟨cifra⟩]$_1^5$

Primedba: Broj reda mora biti iz intervala [1,32767].

Naredba obrade brojnih podataka

$$\left\{\begin{array}{l}\langle\text{brojna promenljiva}\rangle\\ \langle\text{indeksna brojna promenljiva}\rangle\end{array}\right\} = \langle\text{brojni izraz}\rangle$$

Značenje: Vrednost brojnog izraza dodeljuje se promenljivoj na levoj strani znaka jednakosti.

Naredba obrade azbučnih podataka

$$\left\{\begin{array}{l}\langle\text{azbučna promenljiva}\rangle\\ \langle\text{indeksna azbučna promenljiva}\rangle\end{array}\right\} = \langle\text{azbučni izraz}\rangle$$

Značenje: Vrednost azbučnog izraza dodeljuje se promenljivoj na levoj strani znaka jednakosti.

Ograničenje: Ako se azbučna promenljiva sa leve strane znaka jednakosti javlja u azbučnom izrazu na desnoj strani znaka jednakosti, tada se ova promenljiva mora javiti kao prva sleva u azbučnom izrazu. Vrednost koja se dodeljuje promenljivoj može biti najviše dužine 16 znakova.

Naredba dimenzionisanja azbučnog niza

ARR$ (⟨brojni izraz⟩)

Značenje: Celobrojna vrednost brojnog izraza određuje broj elemenata azbučnog niza. Tako, na primer, ARR$ (10) definiše azbučni niz X$ od 11 elemenata: X$(0), X$(1), . . . , X$(10). Svaki elemenat može imati za vrednost azbučni podatak dužine od najviše 16 znakova. Ako je dužina podatka kraća od 16, onda je podatak zdesna omeđen bajtom čija je vrednost (00)$_{16}$.

Naredba upisa bajta u memoriju

BYTE(⟨brojni izraz⟩ , ⟨brojni izraz⟩)

Značenje: Celobrojna vrednost prvog sleva brojnog izraza određuje adresu bajta u memoriji, u koji se upisuje celobrojna vrednost definisana drugim brojnim izrazom.

Naredba za poziv potprograma

CALL ⟨brojni izraz⟩

Značenje: Celobrojna vrednost brojnog izraza definiše broj programskog reda potprograma na BASIC-jeziku na koji se vrši prelazak

---

Naredba postavljanje tačke

DOT ⟨brojni izraz⟩, ⟨brojni izraz⟩

Značenje: Celobrojne vrednosti izraza definišu koordinate tačke na ekranu koja će biti osvetljena Levi gornji ugao na ekranu ima koordinate (0, 0), a desni donji ugao (63, 47).

Naredba za startovanje časovnika

DOT*

Značenje: Naredba startuje časovnik od početne vrednosti koja se postavlja kao vrednost azbučne promenljive Y$, na sledeći način·

Y$="ČČ·MM[·SS [:DD]]"

gde su ČČ — časovi, MM — minuti, SS — sekundi i DD — stoti delovi sekunde, pri čemu se odbrojavanje vrši po dva stota dela sekunde

---

Naredba programskog ciklusa

FOR⟨brojna promenljiva⟩=⟨brojni izraz⟩ TO ⟨brojni izraz⟩[STEP⟨brojni izraz⟩]

Značenje: Naredba FOR definiše početak programskog ciklusa, a kraj je definisan naredbom NEXT Brojna promenljiva se zove indeks ciklusa i pri prvom prolazu kroz ciklus ima vrednost brojnog izraza iza znaka jednakosti, a u svakom sledećem celobrojnu vrednost koja se dobija uvećanjem prethodne vrednosti za priraštaj definisan brojnim izrazom iza reči STEP. Kraj ciklusa

definiše vrednost brojnog izraza iza reči TO. Ako se priraštaj ne navede uzima se da je 1.

Naredba bezuslovnog prelaska

GOTO {⟨brojni podatak⟩ | ⟨brojna promenljiva⟩}

Značenje: Celobrojna vrednost brojnog podatka, odnosno brojne promenljive, određuje broj programskog reda na koji se vrši prelazak

---

Naredba za brisanje ekrana

HOME

Značenje: Naredba briše ekran, ukida zamrzavanje ekrana i pokazivač pozicije na ekranu postavlja na početak prvog reda na ekranu

---

Naredba za zamrzavanje ekrana

HOME ⟨brojni podatak⟩

Značenje: Celobrojna vrednost brojnog podatka određuje poziciju na ekranu iznad koje je ekran zamrznut

---

Naredba uslovnog prelaska (prvi oblik)

IF ⟨relacijski izraz⟩⟨lista naredbi⟩ [:ELSE ⟨lista naredbi⟩]

Značenje: Ako relacijski izraz ima vrednost 1, tada se izvršavaju naredbe navedene u listi naredbi iza relacijskog izraza, u protivnom prelazi se na sledeći programski red, odnosno na listu naredbi iza reči ELSE ako je navedena. Ovde je

⟨lista naredbi⟩: : = ⟨naredba⟩[:⟨naredba⟩]*

Naredba uslovnog prelaska (drugi oblik)

IF ⟨relacijski izraz⟩ [:]ELSE ⟨lista naredbi⟩

Značenje: Ako relacijski izraz ima vrednost 0, tada se izvršavaju naredbe navedene u listi naredbi, u protivnom prelazi se na sledeći programski red

Naredba ulaza

```
INPUT { ⟨promenljiva⟩ | ⟨indeksna promenljiva⟩ }
```

Značenje: Navedenoj promenljivoj dodeljuje se vrednost sa ulaza (tastature).

Naredba za kraj programskog ciklusa

```
NEXT ⟨brojna promenljiva⟩
```

Značenje: Naredba definiše kraj programskog ciklusa, čiji je početak definisan naredbom FOR, a brojna promenljiva mora biti indeks ciklusa.

Naredba izlaza

```
PRINT [ [ { ⟨izraz⟩ | AT⟨brojni izraz⟩ } { , | ; } ] ]* [ [ { ⟨izraz⟩ | AT⟨brojni izraz⟩ } [ , | ; ] ] ]
```

Značenje: Naredba izdaje vrednosti izraza sleva nadesno na ekran, pri čemu celobrojna vrednost izraza iza reči AT definiše poziciju na ekranu od koje počinje izdavanje. Ako između izraza stoji zarez (,) tada se vrednost sledećeg izraza izdaje od početka polja od 8 pozicija na ekranu. Ako su izrazi razdvojeni tačka-zarezom (;), tada se vrednost sledećeg izraza izdaje iza prethodno izdate vrednosti na ekranu. Ako se naredba završava tačka-zarezom tada pokazivač pozicije na ekranu ostaje u tekućem redu. Ako se navede samo reč PRINT tada se izdaje jedan prazan red na ekranu.

Ograničenje: Nije dozvoljeno navoditi opšti oblik azbučnog izraza, već samo azbučni podatak, azbučnu promenljivu, indeksnu azbučnu promenljivu ili funkciju CHR$.

Naredba povratka iz potprograma

```
RETURN
```

Značenje: Naredba vrši povratak iz BASIC-potprograma na naredbu koja sledi iza naredbe CALL kojom je izvršen prelazak na potprogram.

Naredba algoritamskog kraja programa

STOP

Značenje: Naredba prekida izvršavanje programa

Naredba za čitanje programske datoteke

$$\text{TAKE} \begin{Bmatrix} \langle\text{promenljiva}\rangle \\ \langle\text{indeksna promenljiva}\rangle \end{Bmatrix} \left[, \begin{Bmatrix} \langle\text{promenljiva}\rangle \\ \langle\text{indeksna promenljiva}\rangle \end{Bmatrix} \right]^{*}$$

Značenje: Promenljivim sleva nadesno dodeljuju se vrednosti iz programske datoteke i to počev od podatka koji je ukazan pokazivačem programske datoteke

---

Naredba za upravljanje pokazivačem programske datoteke

TAKE ⟨brojni izraz⟩

Značenje: Postavlja pokazivač programske datoteke na naredbu sa izračunatim brojem reda Ovde važi ograničenje: prvi argument brojnog izraza mora biti brojni podatak (može i nula)

TAKE 0

Značenje: Postavlja pokazivač programske datoteke na početak datoteke

---

Naredba za brisanje tačke

UNDOT ⟨brojni izraz⟩, ⟨brojni izraz⟩

Značenje: Celobrojne vrednosti izraza definišu koordinate tačke na ekranu koja će biti ugašena

---

Naredba za zaustavljanje časovnika

UNDOT*

Značenje: Zaustavlja rad časovnika a stanje časovnika može biti pročitano kao tekuća vrednost azbučne promenljive Y

Naredba upisa reči u memoriju

WORD ⟨brojni izraz⟩, ⟨brojni izraz⟩

Značenje: Celobrojna vrednost prvog sleva brojnog izraza određuje adresu registra (dva bajta) u memoriji, u koji se upisuje celobrojna vrednost definisana drugim brojnim izrazom.

Naredba za komentar

! [⟨osnovni simbol⟩]*

Značenje: Naredba omogućuje zapis komentara u programu. Naveden tekst iza uskličnika (!) ne tumači se pri izvršavanju programa.

Naredba za formiranje programske datoteke

⟨podatak⟩ [, ⟨podatak⟩]*

Značenje: Jedna ili više ovih naredbi obrazuju podatke u okviru programa. Azbučni podatak se mora navesti između znakova navoda, a pri tome može sadržavati i zarez (,).

LITERATURA:

[1] Z. Salčić: Mikroračunarski sistemi, Izdavač "Svetlost", Sarajevo, 1982.

[2] N. Parezanović: Računari i programiranja, za III razred srednjeg usmerenog obrazovanja matematičko-tehničke struke, Naučna knjiga, Beograd, 1979.

[3] N. Parezanović: O pisanju čitljivih programa na FORTRAN-jeziku, časopis "PRAKSA", Broj 2, 1984.

[4] O. J. Dahl, E. W. Dijkstra, C. A. R. Hoare: Structured Programming, Academic Press, London, 1972.

[5] V. Stojković, D. Tošić, I. Stojmirović: Programski jezik PASCAL, Naučna knjiga, Beograd, 1984.

[6] R. L. London: The Current State of Proving Programs Correct, Proceedings of 1972 ACM Annual Conference.

[7] N. Parezanović: Računari i programiranje za IV razred usmerenog obrazovanja, Naučna knjiga, Beograd 1980.

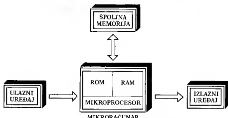
[8] N. Parezanović: Uvod u programiranje, Privredno-finansijski vodič, Beograd, 1983.

[9] D. Ristanović: Računari u vašoj kući, specijalno izdanje časopisa "Galaksija", Januar, 1984.

1
MIKRORAČUNARSKI SISTEM GALAKSIJA

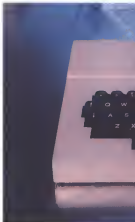
1.1. Konfiguracija sistema

Svaki *računarski sistem* se sastoji od *računara i perifernih uređaja.* Periferni uređaji mogu biti vrlo raznoliki po nameni i karakteristikama. To su uređaji preko kojih računar komunicira sa spoljnim svetom. *Ulazni uređaji* obezbeđuju donošenje informacija iz spoljnjeg sveta u računar, a *izlazni uređaji* obezbeđuju izdavanje informacija iz računara u spoljni svet. Pored ovih uređaja, u periferne uređaje se ubrajaju i *spoljne memorije,* koje služe za čuvanje veće količine informacija (sl. 1.1.). Ako računar sadrži procesor u vidu jedne integrisane komponente (čipa), tada se ovakav procesor zove *mikroprocesor,* a za računar se kaže da je *mikroračunar.* Mikroračunar sa svojim perifernim uređajima čini *mikroračunarski sistem.* Kako na isti računar, odnosno mikroračunar, mogu biti priključeni različiti periferni uređaji to konkretni periferni uređaji u sistemu definišu *konfiguraciju* mikroračunarskog sistema. *Unutrašnja* ili *operativna* memorija mikroračunara se sastoji od ROM (Read Only Memory) i RAM (Random Access Memory) memorije. U ROM memoriji se nalaze sistemski programi, a u RAM memoriji programi korisnika. Kako se sadržaj ROM memorije može samo čitati to su sistemski programi na ovaj način zaštićeni od eventualnih grešaka korisnika [1].



Sl. 1.1. Mikroračunarski sistem

Prof Dr NEDELJKO PAREZANOVIĆ
BASIC ZA GALAKSIJU
Recenzenti
Mr GORDANA PAVLOVIĆ-LAŽETIĆ
DEJAN RISTANOVIĆ
Izdavači
ZAVOD ZA UDŽBENIKE I NASTAVNA SREDSTVA — BEOGRAD
OOUR STVARANJE I PROIZVODNJA NASTAVNIH SREDSTAVA
BIGZ OOUR DUGA
ĆASOPIS ZA POPULARIZACIJU NAUKE GALAKSIJA
Glavni i odgovorni urednici
ALEKSANDAR ZONJIĆ
GAVRILO VUĆKOVIĆ
Urednici
SVETISLAV TODIĆ
JOVO REGASEK
Likovno grafička obrada
DRAGUTIN NJEŽIĆ
Naslovna strana
DRAGUTIN NJEŽIĆ
MILAN BJELANOVIĆ
Tiraž 3 000 primeraka
decembar 1984
Štampa OOUR »Radiša Timotić«, Jakšićeva 9, Beograd



Prof. dr Nedeljko Parezanović

BASIC ZA GALAKSIJU



PREDGOVOR

Mikroračunar GALAKSIJA pripada klasi *personalnih* računara i može se koristiti za ličnu upotrebu, ali takođe i u obrazovanju, kao i u raznim drugim prilikama u kojima personalni računari sve više nalaze primenu. Zapravo, vrlo je teško govoriti o granici mogućnosti određene klase računara u primenama. Ovo u mnogome zavisi od sposobnosti korisnika da iskoristi mogućnosti računara na najbolji način. Za prenošenje zadataka na računar korisniku stoje na raspolaganju, kod raznih računara, različiti jezici programiranja. Za računar GALAKSIJU korisniku je na raspolaganju BASIC-jezik. Ova knjiga je pisana kao priručnik za programiranje na BASIC-jeziku za računar GALAKSIJU. Pri pisanju ovakvih priručnika treba zadovoljiti dva oprečna uslova:

- navesti precizne definicije svih konstrukcija jezika i
- primeniti postupnost u izlaganju materije.

Prvi uslov proističe iz potrebe preciznog navođenja i opisa svih mogućnosti jezika, a drugi iz potrebe upotrebe priručnika od strane što šireg kruga korisnika. Međutim, postupnost u izlaganju otežava precizan prikaz konstrukcija jezika, a priručnik uz računar treba pre svega ovaj uslov da zadovolji. Ovaj priručnik je pisan sa namerom da što više zadovolji oba navedena uslova. Međutim, ipak se pretpostavlja da korisnik poznaje problem algoritmizacije zadataka na nivou algoritamskih šema, programiranje na programskim jezicima često i ne zahteva pisanje algoritamskih šema [7].

Autor ovog teksta veruje da će GALAKSIJA naći mesto u školama, pre svega za potrebe obrazovanja iz oblasti programiranja. Zato je ovaj priručnik po strukturi i načinu izlaganja blizak udžbeniku koji se koristi u srednjem usmerenom obrazovanju [6]. Na ovaj način, nastavnici i učenici mogu bez posebnih teškoća koristiti priručnik, s tim što se pretpostavlja da pri čitanju priručnika raspolažu i mikroračunarom GALAKSIJA.

Časopis »Galaksija« je mnogo doprineo širokoj popularizaciji mikroračunara GALAKSIJA, posebno preko svog specijalnog izdanja [9]. Ovaj priručnik će svim korisnicima koji su upoznali popularno prikazan BASIC jezik omogućiti da upoznaju precizno izložene konstrukcije jezika, kao i da se sretnu sa više rešenih zadataka i zadataka za vežbu. Pored toga, priručnik je pisan tako da korisnik što više upozna koncepciju u izgradnji BASIC jezika, što će omogućiti korišćenje i drugih knjiga iz oblasti programskih jezika. Na kraju knjige, u dodacima, je izložena stroga definicija sintakse BASIC-jezika za računar GALAKSIJU, i ukratko objašnjena semantika pojedinih konstrukcija. Ovo će omogućiti korisnicima koji imaju programersko iskustvo da brzo upoznaju mogućnosti BASIC-a za GALAKSIJU, a ostalima da lako dobiju uvid u sve konstrukcije jezika i njihove uzajamne veze.

Autor će biti zahvalan svim čitaocima koji svojim primedbama doprinesu da eventualna buduća izdanja ove knjige budu poboljšana u stručnom ili terminološkom smislu.

13. april 1984.
Beograd

Autor

SADRŽAJ